FISH SPECIES IDENTIFICATION USING IMAGE ANALYSIS OF ECHO-SOUNDER IMAGES

PATRICIA LEFEUVRE

FISH SPECIES IDENTIFICATION USING IMAGE ANALYSIS OF ECHO-SOUNDER IMAGES

By

© Patricia LeFeuvre, B. Eng.

A thesis submitted to the
School of Graduate Studies
in partial fulfilment of the
requirements for the degree of
Master of Engineering

FACULTY OF ENGINEERING AND APPLIED SCIENCE
MEMORIAL UNIVERSITY OF NEWFOUNDLAND
December, 2002

ST. JOHN'S NEWFOUNDLAND CANADA

Abstract

Acoustic surveys for marine fish in coastal waters typically involve identification of species groups. Incorrect classification can limit the usefulness of both distribution and biomass estimates. Fishing catch data can assist in identification, but are rarely spatially comparable to acoustic data and are usually biased by gear type. This thesis describes a technique and a software toolkit, "FASIT" (Fisheries Assessment and Species Identification Toolkit), developed by the author to enable automated identification of Atlantic cod (*Gadus morhus)*, capelin (*Mallotus villosus*), and redfish (*Sebastes spp.*) based on high resolution acoustic imaging of fish aggregations. The approach has been to assess and analyze various amplitude, shape and location features of the acoustic returns from shoals and individual fish, then to use these features to develop algorithms which discriminate among species. Fourteen classifiers based on Three-Nearest Neighbour classification and Mahalanobis distance classification have been implemented and tested. The best classifier had an average correct classification rate of 96.8%. The data used for this thesis are fisheries data from a number of Newfoundland bays and the Grand Bank region collected using a 38 KHz digital echo-sounder.

Acknowledgments

I would like to thank the following institutions for funding this project: OPEN (NSERC funded network of excellence in fisheries); the Canadian Centre for Fisheries Innovation; the NSERC Fisheries Conservation Chair; the Department of Fisheries and Oceans; C-CORE; and Memorial University of Newfoundland's Faculty of Engineering and Applied Science. I would also like to thank George Rose, Rodney Hale, Ray Gosine, and John Guzzwell.

Table of Contents

Abstract ii

Lists of Figures vi

List of Tables viii

List of Symbols, Abbreviations, and Acronyms ix

1.0 INTRODUCTION 1

2.0 BACKGROUND 4
- 2.1 Echo-sounder Technology 4
- 2.2 Biological Acoustics 9
  - 2.2.1 Physiology 10
  - 2.2.2 Behaviour 14

3.0 REVIEW OF FISHERIES ACOUSTICS LITERATURE 17
- 3.1 Taxonomic Identification of Fish Species Publications 17

4.0 DATA COLLECTION AND ANALYSIS 25
- 4.1 Acoustic Data Collection 25
- 4.2 Data Pre-processing 29
- 4.3 Segmentation 30
- 4.4 Feature Extraction 35
- 4.5 Classifier Design 42
  - 4.5.1 Feature Reduction 42
    - 4.5.1.1 Biased Feature Removal 43
    - 4.5.1.2 Redundant Feature Removal 44
    - 4.5.1.3 Classifier Specific Feature Reduction 46
  - 4.5.2 Classifier Types 48
    - 4.5.2.1 Three Nearest-Neighbor Classifier 48
    - 4.5.2.2 Mahalanobis Distance Classifier 49
  - 4.5.3 Classifier Configurations 51

5.0 RESULTS AND DISCUSSION 53
- 5.1 Classifier Performance 53
  - 5.1.1 Three-Nearest Neighbor Classifiers 53
    - 5.1.1.1 Classifier Configuration #1 53
    - 5.1.1.2 Classifier Configuration #2 54

5.1.1.3 Classifier Configuration #3 55
5.1.1.4 Classifier Configuration #4 56
5.1.2 Mahalanobis Distance Classifiers 57
5.1.2.1 Classifier Configuration #5 57
5.1.2.2 Classifier Configuration #6 57
5.1.2.3 Classifier Configuration #7 58
5.1.2.4 Classifier Configuration #8 59
5.1.3 Combination Classifiers 60
5.1.3.1 Classifier Configuration #9 60
5.1.3.2 Classifier Configuration #10 61
5.1.3.3 Classifier Configuration #11 62
5.1.3.4 Classifier Configuration #12 63
5.1.3.5 Classifier Configuration #13 64
5.1.3.6 Classifier Configuration #14 64
5.2 Discussion 65

6.0 CONCLUSION 70

7.0 RECOMMENDATIONS 72

8.0 REFERENCES 74

Appendix A Data Collection Area Maps 79

Appendix B Acoustic Calculations 82

Appendix C Echo-sounder Parameters 85

Appendix D Matlab© Code 87

BIBLIOGRAPHY

Lists of Figures

Figure 1: Echo-sounder graphic 5

Figure 2: Beam pattern measurements for a 120 kHz BioSonics DT transducer 6

Figure 3: Example of an echogram 9

Figure 4: Dorsal aspect reflectivity pattern for cod and saithe 12

Figure 5: Derivation of the hyperbolic arc 13

Figure 6: Theoretical hyperbolic arcs for fish with different reflectivity patterns 14

Figure 7: Effect of movement on single target hyperbolic arcs 16

Figure 8: FASIT echogram illustrating distributions of schooling capelin 26

Figure 9: FASIT echogram illustrating distributions of densely schooling cod 27

Figure 10: FASIT echogram illustrating single cod 27

Figure 11: FASIT echogram illustrating single and small groups of redfish 28

Figure 12: Bottom detection block diagram 31

Figure 13: Image processing block diagram 32

Figure 14: a) Original greyscale "image"after bottom removal, (b) After threshold operation, (c) After morphological operations 33

Figure 15: Labeled image 34

Figure 16: Feature plot of $D_{centroid}$ and D_{top} (in meters) 43

Figure 17: Feature plot showing *El* and *Comp* 45

Figure 18: Feature plot showing CMY2 and BCMY2 45

Figure 19: Single node 3NN classifier accuracy as a function of the number of

features used 47

Figure 20: Single node Mahalanobis classifier accuracy as a function of the number of features used 47

Figure 21: Single-node classifier 51

Figure 22: Dual node classifier tree structure 51

Figure 23: Feature plot for *Comp* and *W (meters)* 68

Figure 24: Feature plot for *BCM2* and *BCMX1Y1* 68

Figure 25: Feature plot for Amp_{mean} (dB) and Amp_{sd} 69

List of Tables

Table 1: Echo ability of cod, capelin and redfish at 38 KHz 11

Table 2: Typical lengths of mature cod, capelin and redfish 11

Table 3: Estimated *TS* range for mature cod, capelin and redfish 11

Table 4: Mean feature values for each species 50

Table 5: Classifier combinations 52

Table 6: Confusion matrix for classifier configuration #1 54

Table 7: Confusion matrix for classifier configuration #2 55

Table 8: Confusion matrix for classifier configuration #3 56

Table 9: Confusion matrix for classifier configuration #4 56

Table 10: Confusion matrix for classifier configuration #5 57

Table 11: Confusion matrix for classifier configuration #6 58

Table 12: Confusion matrix for classifier configuration #7 59

Table 13: Confusion matrix for classifier configuration #8 60

Table 14: Confusion matrix for classifier configuration #9 61

Table 15: Confusion matrix for classifier configuration #10 61

Table 16: Confusion matrix for classifier configuration #11 62

Table 17: Confusion matrix for classifier configuration #12 63

Table 18: Confusion matrix for classifier configuration #13 64

Table 19: Confusion matrix for classifier configuration #14 65

Table 20: Summary of classifier results 67

List of Symbols, Abbreviations, and Acronyms

Symbol	Description
α	Absorption coefficient (dB / meter)
σ	Backscattering cross section (centimeters^2)
τ	Pulse length (meters)
λ	Acoustic wavelength (meters)
A	Area (meters^2)
Amp_{max}	Max amplitude (dB relative to the return from a 2 m diameter sphere)
Amp_{mean}	Mean amplitude (dB relative to the return from a 2 m diameter sphere)
Amp_{min}	Min amplitude (dB relative to the return from a 2 m diameter sphere)
Amp_{sd}	Amplitude standard deviation
Ax	Axis (degrees)
BCMX1Y1	Binarized Central Moment X1Y1
BCMX2	Binarized Central Moment X2Y0
BCMY2	Binarized Central Moment X0Y2
Comp	Compactness
c	Sound velocity (meters / second)
CMX1Y1	Central Moment X1Y1 (dB relative to the return from a 2 m diameter sphere)
CMX2	Central Moment X2Y0 (dB relative to the return from a 2 m diameter sphere)
CMY2	Central Moment X0Y2 (dB relative to the return from a 2 m diameter sphere)
D	Target depth (meters)
d	Characteristic linear size of a target defined as the cube root of its volume (centimeters)
$D_{centroid}$	Depth to the centroid of the object (meters)
D_{top}	Depth to the top of the object (meters)

$Dist_{bottom}$	Distance from the object to the seabed (meters)
El	Elongation
f	Pulse carrier frequency or sonar operating frequency (Hz)
$FeretEl$	Feret elongation
H	Height (meters)
I_i	Incident sound intensity (power / unit area)
I_r	Reflected sound intensity (power / unit area)
k	Size of bottom detection window (sample points)
L	Fish length (centimeters)
m	Median filter length (pings)
$MnFeret$	Minimum Feret Diameter (meters)
$MxFeret$	Maximum Feret Diameter (meters)
n	Number of sample points in a ping
P	Perimeter (meters)
p	Sample point
R	Roughness
r	Range (meters)
t	Time delay (seconds)
TS	Target Strength (dB)
TVG	Time Varied Gain
W	Width (meters)

1.0 INTRODUCTION

Species identification is "the grand challenge of fisheries and plankton acoustics" (MacLennan and Holliday, 1996). Fishermen and the military accepted this challenge early on, and many can identify the "marks" on their echo-sounders with good success. In marine science, hydroacoustic biomass surveys have depended on concurrent fishing to identify the fish species observed. Problems with concurrent fishing are primarily due to two factors: the variability in the species' catchability and the difficulty in achieving spatial or temporal sampling comparable with that of acoustic sampling.

The first formal attempts to classify the myriad of pulses and shapes that appear in the water column on echosounders, and to identify them with some certainty, were made in the late 1970's and early 1980's (Holliday, 1977; Deuser et al., 1979; Giryn et al., 1979; Zakharia and Sessarego, 1982). These attempts were followed by several empirical studies that applied simple signal processing techniques to fisheries acoustic data (Rose and Leggett, 1988; Souid, 1988; Diner et al., 1989). The availability of inexpensive and high speed small computers in the late 1980's, coupled with the minor successes of the earlier attempts to classify fish echoes to species, spurred research using narrow-band (single acoustic frequency) systems. Attempts to use the information from single echos (or pings) gave way to image processing techniques capable of assessing many pings at once, as complete images (Weill et al., 1993; Lu and Lee, 1995; Reid and Simmonds, 1993; Richards et al., 1991). Several of these methods provided high rates of correct classification in restricted ecological

situations, but none have provided a classifier which is successful over broad ranges of time and space (see review by Scalabrin et al., 1996). Several recent efforts have been made to use wide-band acoustics for classification (Simmonds and Armstrong 1990; Simmonds et al., 1996; Zakharia et al., 1996). These methods show considerable promise in experimental studies but they require equipment which for now is well out of the budgetary reach of most fisheries organizations.

For some marine echo-systems, especially those at high latitudes (like Newfoundland) where the number of different fish species is low it appears that information from narrow-band echosounders (like the system used for this study) may suffice for classification. It is important to recognize that it is unlikely that any classification algorithm can be developed to classify all species over a broad range of ecological conditions (Scalabrin et al., 1996). Rather, to increase the probability of success, it is necessary to develop knowledge of the system under study, and to limit the questions to be resolved and species to be classified. This detracts little from most applications.

In Newfoundland coastal waters, the most common species encountered on an echosounder are Atlantic capelin (*Mallotus villosus*), herring (*Clupea harengus*) and cod (*Gadus morhua*). In some areas Atlantic mackerel (*Scomber scombrus*) and redfish (*Sebastes spp.*) are commonly observed. There is a great deal of seasonal variation in distribution and aggregation patterns in all these species. During research and surveys it is imperative that the acoustic traces from these species be consistently identified with a high degree of

accuracy. Hence, one aspect of this research was initiated to develop methods to extract information from high-resolution digital backscatter that might lead to improved signal classification. Another aspect of this research was the exploration of a number of different classification techniques for application to this information.

The specific objective of the research described here was the development of an algorithm for timely classification of Atlantic capelin (*Mallotus villosus*), cod (*Gadus morhua*) and redfish (*Sebastes spp.*) using image processing techniques and pattern recognition. The data used for this research was collected from Placentia Bay, Newfoundland, Trinity Bay, Newfoundland, and the 3Ps region of the Grand Bank [see Appendix A for maps].

To make the results of the research easily usable, a Windows-based software application, known as FASIT (Fisheries Assessment and Species Identification Toolkit) has been developed. FASIT is used for post-processing of fisheries acoustic data. It can perform biomass estimation using echo integration and species identification for capelin (*Mallotus villosus*), cod (*Gadus morhua*) and redfish (*Sebastes spp.*). The version of the FASIT program described in this thesis was developed by the author.

Much of the work described in this thesis has also been published in Fisheries Research by LeFeuvre et al. in an article entitled "Acoustic species identification in the Northwest Atlantic using digital image processing."

2.0 BACKGROUND

The literature describing underwater acoustics is extensive. Elementary principles have been described very well in Clay and Medwin, 1977 and Urick, 1983. This Section will therefore be dedicated to describing aspects of underwater acoustics that are particularly relevant to the problem of fish species identification. Section 2.1 gives a brief technical introduction to echo-sounder technology for readers unfamiliar with fisheries acoustics. Section 2.2 describes some aspects of fish as acoustic targets that contribute to making species identification possible.

2.1 Echo-sounder Technology

A fisheries echo-sounder is a SONAR (SOund NAvigation and Ranging) system which transmits an acoustic signal (or ping), most often in a vertical direction toward the seabed [Figure 1 A]. The most common type of fisheries echo-sounder is single-beam, single-frequency. The transmitted signal emitted by the transducer (typically a piezoelectric crystal) is generally a pulsed (duration τ) single-frequency (f) sinusoid of constant amplitude. In fisheries science most work has historically been done using short pulse lengths, usually from 0.2 msec to 1.0 msec, and a limited number of carrier frequencies, primarily 38KHz and 120 KHz (Johannesson and Mitson, 1983). These parameters provide a good compromise between signal range and signal resolution, and by using standard parameters researchers

have been able to make use of other's work.

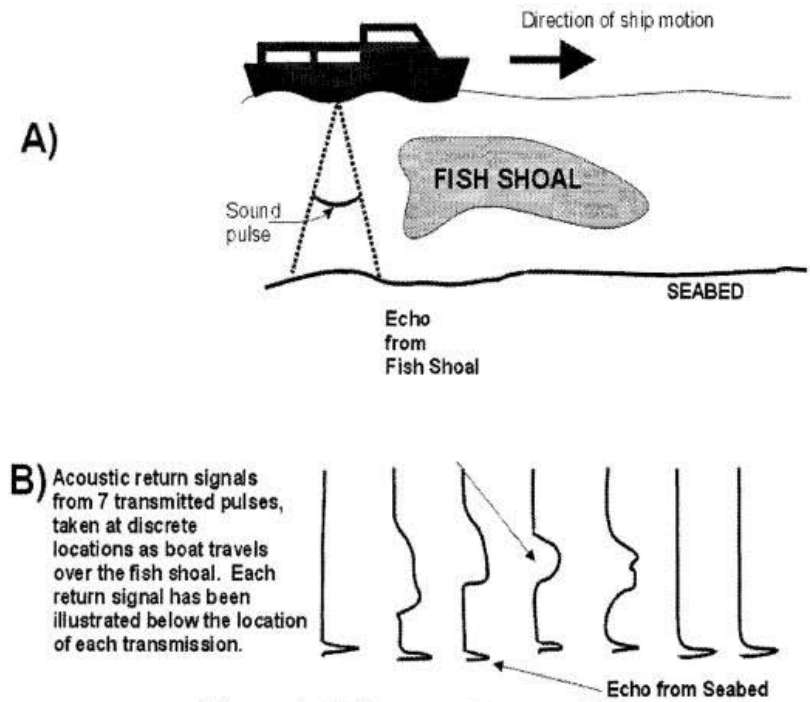


Figure 1: Echo-sounder graphic

An acoustic pulse is a mechanical disturbance that propagates as a pressure wave in a directional beam pattern away from the transmitter. The 3dB beamwidth of a typical fisheries echo-sounder is between 5 and 15 degrees (MacLennan and Simmonds, 1992). Figure 2 illustrates the beam pattern of a 120 KHz BioSonics DT echo-sounder. Due to spherical spreading the intensity of the pressure wave decreases inversely with the square of the distance travelled (r). Any objects located in the transmitted signal's path that have a density not equal to the density of the surrounding water create echos that are returned to the

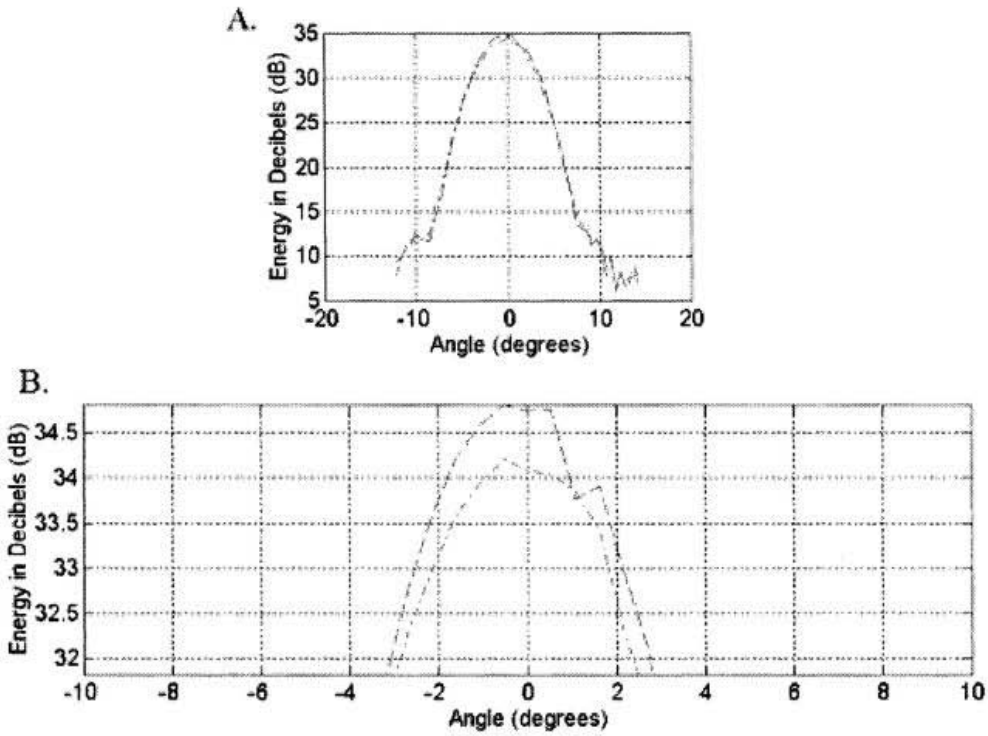


Figure 2: Beam pattern measurements for a 120 kHz BioSonics DT transducer. A. Average energy curve for orthogonal directions; B. Expanded view cut-off 3 dB down from the peak of the taller curve. (LeFeuvre et al,1996)

echo-sounder's receiver (a piezoelectric crystal, commonly the same one used for signal transmission) which transforms the acoustic signal to a proportional voltage signal [like the voltage signals illustrated in Figure 1 B]. To compensate for the spherical spreading losses, a time varied gain (*TVG*) is applied to the returned signal. In fisheries acoustics 40 log(r) *TVG* is commonly applied to signals from single fish while 20 log(r) *TVG* is applied to signals from densely packed fish schools. To compensate for absorption losses (where acoustic energy is converted into heat energy) an appropriate absorption coefficient (α) is applied to the returned signal. The absorption coefficient is in units of dB/m and is constant

for a given acoustic frequency, water temperature and salinity. See Appendix B for the equation used to estimate α.

The distance from the echo-sounder to the reflective object can be calculated using the equation: $r = \frac{1}{2} ct$, where c is sound velocity and t is the time delay between the transmitted pulse and the received echo (MacLennan and Simmonds, 1992). See Appendix B for the estimation of c as a function of water temperature and salinity.

Target strength (TS) is a common way of expressing an object's ability to produce an echo (Johannesson and Mitson, 1983) and is defined by the ratio of the reflected energy (I_r) from a target were it located at a distance of one meter from the sonar, over the incident sound intensity (I_i):

$$TS = 10 \log (I_r/I_i) \text{ in units of dB} \quad \textbf{(1)}$$

The strength of an echo reflected by an object is related to a number of factors including the strength of the incident sound wave, the change in acoustic impedance between the water and the object, and the shape and the size of the object. The greater the strength of the incident sound wave or change in acoustic impedance the greater the echo. The effect of shape and size however is more complicated.

Objects that are very small compared to the wavelength of the incident wave (λ) will act as an acoustic point source of scattered waves which radiate spherically in all directions.

According to the Rayleigh scattering law, the scattered energy is proportional to $(d/\lambda)^4$ when $d<<\lambda$ where d is the characteristic linear size of the target defined as the cube root of its volume (MacLennan and Simmonds, 1992). The strength of echoes from objects that are very large compared to the wavelength of the incident wave is not a function of frequency. The scattering energy from a large spherical object increases approximately as the square of the sphere radius (MacLennan and Simmonds, 1992). For objects with sizes similar to the wavelength of the incident wave, the scattering is related to the shape of the object and its material properties. In this region, resonances can occur as well, making theoretical prediction of scattering strength difficult (MacLennan and Simmonds, 1992).

The returned signals from the echo-sounder transducer are usually displayed as an echogram. Echograms, are graphical displays of the recorded reflection energy from subsequent echos (or pings) taken as the vessel on which the echo-sounder has been mounted traverses along the water. On an echogram subsequent echos are plotted next to each other vertically. Each sample point is plotted using a colour or shade of grey to represent the intensity of the echo received. This generates a two dimensional "image" of a the water column under the path of the vessel. Figure 3 illustrates an echogram displayed using the FASIT software. It is from these echograms that many experienced fishers and fisheries scientists can visually identify the species of fish being displayed.

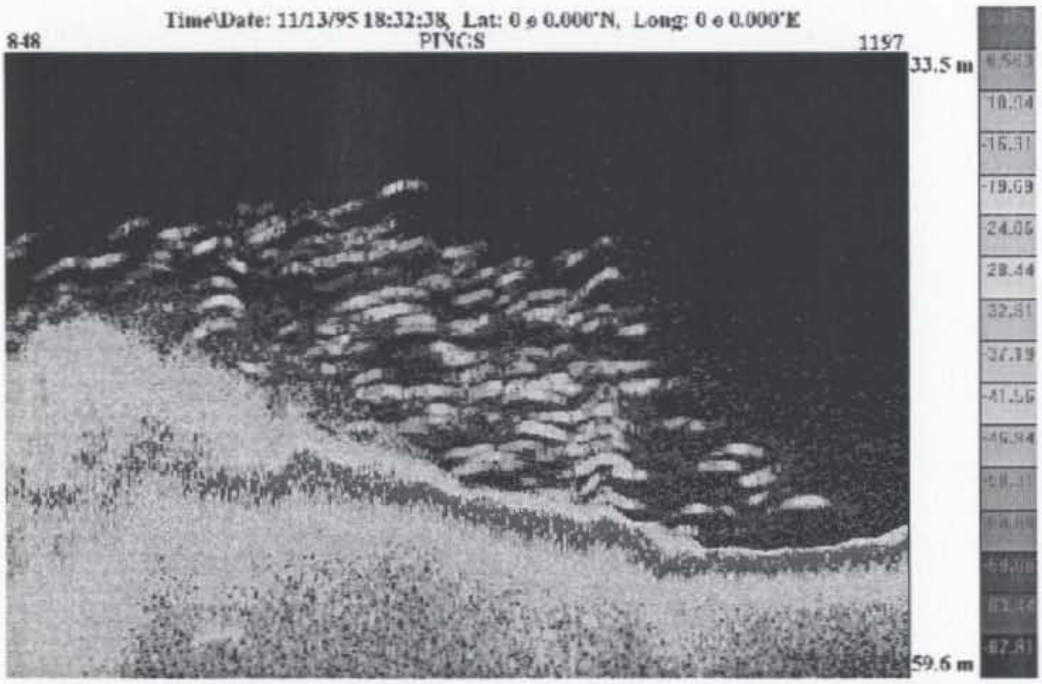


Figure 3: Example of an echogram containing loosely schooled cod located near the seabed

2.2 Biological Acoustics

Fish are neither uniformly nor randomly distributed in the ocean. Different species typically aggregate where different environmental conditions such as depth, temperature, and bottom type occur. Species also aggregate according to oceanographic conditions and time of day or year (Lee et al., 1996). This variation in behaviour among different fish species as well as differences in physiology are what make species identification using acoustic signals possible. Section 2.2.1 will describe some of the physiological differences between species

and how they assist in making identification possible. Section 2.2.2 will describe some behavioural differences which also aid in acoustic identification.

2.2.1 Physiology

The backscattering ability, or target strength, of fish varies from species to species. It is dependent on acoustic frequencies and is related to the physiological characteristics of the species, particularly on whether or not the organisms contain gas (i.e. a swim bladder) (Nakken, 1998). The swim bladder is the major cause of scattering from a bladder-bearing fish contributing anywhere from 90 to 95% of the total echo (Foote, 1980). This is because the acoustic impedance of the gas in the swim bladder is very different from the surrounding water and other organs within a fish (Nakken, 1998).

Within a species the target strength is strongly related to the length of the fish (Love, 1971). In Table 1 the target strength of the three species of fish under study for this thesis are presented as a function of fish length (L) expressed in centimeters. As shown, there are differences between the three species. The echo from a cod is approximately 7.1dB greater than that of an equally long capelin and 2.1dB greater that of an equally long redfish. Table 2 lists the typical length range of mature fish for the species of interest. Please note that typical mature fish lengths vary from year to year, study to study, and region to region, therefore these ranges are used for illustrative purposes only.

Table 1: Echo ability of cod, capelin and redfish at 38 KHz

Species	Target Strength (dB)
Cod (*Gadus morhua*)	20 log(L) - 66*
Capelin (*Mallotus villosus*)	20 log(L) - 73.1**
Redfish (*Sebastes marinus*)	20 log(L) - 68.1***

* Rose and Porter, 1996, ** Rose, 1999, *** Gauthier and Rose, 2001

Table 2: Typical lengths of mature cod, capelin and redfish

Group	Typical mature lengths (cm)
Cod (*Gadus morhua*)	> 45*
Capelin (*Mallotus villosus*)	> 12 *
Redfish *(Sebastes spp.)*	> 24 *

* Correspondence with Dr. George Rose.

With the information from Tables 1 and 2, the expected *TS* range for each species has been calculated and is given in Table 3. Clearly, for identification of these three species, echo strength data will provide very important classification information.

Table 3: Estimated *TS* range for mature cod, capelin and redfish

Group	Estimated *TS* Range (dB)
Cod (*Gadus morhua*)	> -32.9
Capelin (*Mallotus villosus*)	> -51.5
Redfish *(Sebastes spp.)*	> -40.5

Another physiological factor that may in assist species identification is the variation in swim bladder shape from species to species. Experiments on tethered fish have shown that echo strength (or ability) is dependent on the angle between the fish and the incident sound (tilt angle) (Nakken, 1998). Figure 4 shows the dorsal aspect reflectivity pattern for two gadoid species, cod and saithe (data for capelin and redfish were not available in the literature so these data are being used for illustration only). Note that in this figure, echo ability has been expressed as backscattering cross section (σ) in units of centimeters. Backscattering cross section is related to target strength (*TS*) as follows:

$$TS = 10\log10(\sigma/4\pi). \qquad \textbf{(2)}$$

Both species in Figure 4, cod and saithe, have their maximum echo ability when tilted with their heads down a few degrees but the echo ability of saithe decreases more rapidly with tilt

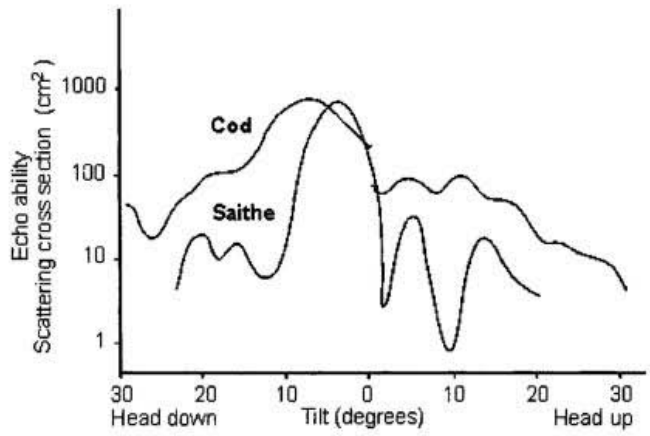


Figure 4: Dorsal aspect reflectivity pattern for cod and saithe (Nakken, 1998)

angle than it does for cod. The reason for this is the more elliptical or spherical shape of the cod swim bladder and the more elongated cylindrical shape of the saithe swim bladder (Midttun and Hoff, 1962).

It is expected that these reflectivity patterns will produce different characteristic hyperbolic arcs for each species. A hyperbolic arc appears on an echogram when an individual fish is insonified by more than one ping. A reflection of an object is plotted on an echogram as if the object was positioned directly beneath the sonar device, independent of its point of origin within the transducer beamwidth [see Figure 5]. For fish targets, there

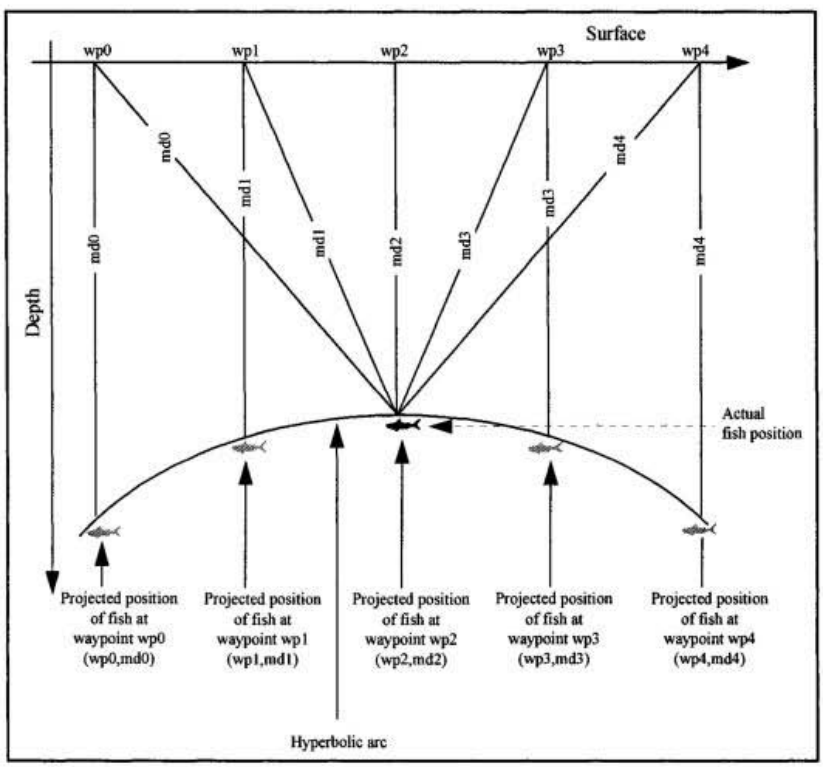


Figure 5: Derivation of the hyperbolic arc resulting from plotting a single point whose energy is spread over several pings.

is a significant difference between the actual "point" fish target and the recorded hyperbolic reflection event. Figure 5 graphically illustrates how a point target is transformed to a hyperbola when the acoustic returns are plotted on an echogram. In this figure, reflected energy from the fish target at location (wp2,md2) appears at five different locations.

Figure 6 illustrates the difference a fish's reflectivity pattern could theoretically have on a the resulting hyperbolic arc. The object illustrated in 6 (a) is a modelled hyperbolic arc given a fish with a more directional reflectivity pattern than the fish used to model the arc in 6 (b). The simulated arcs were generated for two fish with the same maximum echo ability but different reflectivity patterns (as was the case for cod and saithe in Figure 4). As shown, the two arcs have different shapes. Given this, it may be possible to discriminate between fish with equal *TS* if they have different reflectivity patterns.

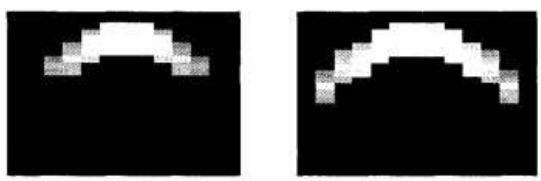


Figure 6: Theoretical hyperbolic arcs for fish with different reflectivity patterns

2.2.2 Behaviour

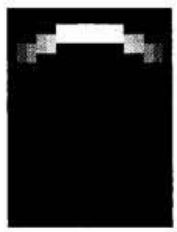
Discrimination between species is also aided by differences in fish behaviour. Fish behaviour has been studied for over 35 years with the aid of echo-sounder technology and

acoustic tags (Misund, 1997). Many studies have revealed species behaviour patterns that may directly or indirectly aid in remote species identification (Misund, 1997 summarizes a number of interesting studies). For example, many species show clear preferences for swimming depth or off bottom distance. Some species prefer very specific temperatures, therefore they are often found in thermal "layers" in the water column. As a result, depth and off bottom distance features can be helpful for species identification. A number of species exhibit what is known as avoidance behaviour: they avoid moving ships (Misund, 1997). Rapid swimming or diving away from a survey vessel could alter the shape of the resulting arc in the echogram as illustrated in Figure 7.

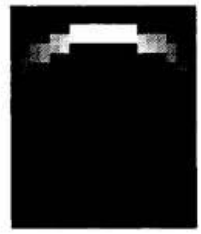
Other behaviours such as schooling can provide another set of features that are helpful for species identification. When fish school their hyperbolic arcs are no longer visible in an echogram. It has been shown, however that the amplitudes of echos from schooling fish are related to the number of fish within the acoustic beam i.e. the schooling density (MacLennan, 1992). Schooling densities as well as school shapes, sizes and location in the water column are characteristic for different species (Misund, 1997).

Some commonly recognized behaviours of the species of interest for this thesis are as follows. Capelin are typically found in schools year round, although the size and density of the schools vary depending on the time of year, the time off day, the tides, and other factors (Jangaard, 1974). These schools are usually located midwater or near the surface (Rose and Leggett,1988). Redfish tend to stay close to the seabed during the day, moving

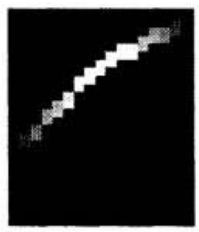
upward at night to feed (Pikanowski, 1999). Cod are typically found alone and near the seabed or in very dense aggregations especially during spawning (Rose, 1992).



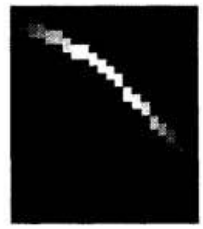
A. Stationary target



B. Moving in the direction of the boat



C. Moving up and horizontally in the direction of the boat



D. Moving down and horizontally in the direction of the boat

Figure 7: Effect of movement on single target hyperbolic arcs

3.0 REVIEW OF FISHERIES ACOUSTICS LITERATURE

A literature review has been conducted on fisheries acoustics in general and more specifically on taxonomic identification of fish using acoustic signals. A list of helpful papers and books reviewed but not specifically mentioned in the following Sections can be found in the Bibliography. The literature regarding fisheries acoustics in general has been discussed in Section 2.0. Section 3.1 will summarize the literature specifically describing taxonomic identification of fish species.

3.1 Taxonomic Identification of Fish Species Publications

Early attempts at automated fish identification involved detailed analysis of the echo signal (Giryn *et al.* 1981, Rose and Leggett 1988, and Magand and Zakharia 1992). Giryn, Rojewski, and Somla (1981) describe a method to identify 'sea creature species' on the basis of their hydroacoustic echo signals. They calculated the central moments of individual echoes, which roughly determined a probability density function, and used them as inputs to a Euclidean distance classifier. The paper briefly describes tests of their recognition system carried out on echoes from (1) horse mackerel (*Decapterus macrosoma*) schools, (2) single-species single-fish layers, (3) single-species multiple scattering layers, and (4) the sea bottom. The paper did not provide classification results but did state that the system operated with virtually no classification errors and that the classification of different fish species would be possible in the future.

Rose and Leggett (1988) took two approaches to species identification: (1) using target strength measurements of individual fish and (2) using features of the backscattered energy from schooling fish. In the study by Rose and Leggett, three species were tested: cod (*Gadus morhua*), capelin (*Mallotus villosus*), and mackerel (*Scomber scombrus*). As mentioned in Section 2.2.1, fishes without swim bladders, such as mackerel, have target strengths well below those of even much smaller fishes with swim bladders such as capelin (Nakken and Olsen, 1977). Using a 120 KHz transducer Rose and Leggett measured the relationship between fish length and target strength for each species and their results were in agreement with those of various previous studies (Nakken and Olsen, 1977, Midttun, 1984, and Foote 1987). While the relationship between length and target strength differed for each species, the hypothesis that target strength alone could be used for classification was proven false. First of all, the target strength of large mackerel was similar to the target strength of capelin. The second reason was that the schooling behaviour of the fish under study created fish densities at which the selection of single fish echoes became a very subjective process. As a result, target strength became unpredictable and dependent on the packing density of the school and other behavioural patterns. Rose and Leggett did indicate that target strength could be used for classification if the target species had discrete target strength distributions and when their schooling behaviour allowed for isolation of single targets. They had more success with classification using school descriptors. The following features were extracted from two sequences of four or five pings within each school: (1) off bottom distance, (2) school depth, (3) mean squared voltage, (4) standard deviation of voltage squared, (5) maximum squared voltage, (6) mean distance between within school

voltage peaks (referred to as PP) and (7) mean peak to trough distance of the voltage squared standardized to the mean squared voltage (referred to as SPT). The two most powerful features were PP and SPT which the authors believe reflect internal school structure. Using quadratic classification functions 91% of the 23 capelin schools, 96% of the 26 cod schools and 91% of the 11 mackerel schools were correctly classified.

Later attempts at fish species identification have incorporated analysis of the echo signal and analysis of the two dimensional spatial information in the echogram image (Richards *et al.* 1991, Scalbrin *et al.* 1994, and Lu and Lee 1995). Richards *et al.* (1991) report on a project to classify fish schools based on echo integration survey data, in order to demonstrate that typical echo-integration data could be applied in species recognition. They studied schools of rockfish (*Scorpaenidea*) living in two types of habitats. One category stayed in an area of bedrock outcrops, while the other stayed close to a continental slope. The characteristics used to discriminate between the different types of schools were (1) time of day, (2) mean volume density, (3) dispersion, and (4) mean off-bottom distance. Using nearest-neighbour classification they were able to classify the different shoals (105 of them in total) with up to a 97% success rate.

Scalabrin *et al.* (1994) describe the MOVIES-B software developed to perform automated shoal recognition. Their linear discriminant classifier used the following morphological shoal descriptors: (1) length, (2) area, (3) fractal dimension, and (4) elongation, the following bathometric descriptors: (5) bottom depth, (6) shoal depth, and the

following amplitude descriptors:(7) volume reverberation index, (8) average amplitude, and (9) standard deviation of amplitude. Their system was developed using a data set made up of 178 sardine (*Sardina pilchardus*) shoals, 449 anchovy (*Engraulis encrasicolus*) shoals, 645 horse mackerel (*Trachurus trachurus*) shoals, and 93 blue whiting (*Micromesistius poutassou*) shoals. Training on 70% of their available data and testing on the remaining 30%, they were able to discriminate between sardine and blue whiting shoals 100% of the time, between sardine and anchovy 96% of the time, and between blue whiting and anchovy 97% of the time. Their ability to perform classification was reduced when trying to discriminate between other species and horse mackerel: 64% for anchovy, 76% for sardine, and 96% for blue whiting. The authors did not design a classifier to perform classification of all species studied.

Scalabrin et al. (1996) describe further attempts to discriminate between sardine (*Sardina pilchardus*) shoals, anchovy (*Engraulis encrasicolus*) shoals, and horse mackerel (*Trachurus trachurus*) shoals. Although the results quoted were not as successful as the results quoted in the 1994 paper, Scalabrin et al. have described some alternative approaches for species classification. The amplitude probability density function (PDF) approach used shoal PDFs to distinguish between species. The PDFs illustrated differences between anchovy and horse mackerel shoals, but were not strong enough to be used alone for classification. A second approach made use of spectral descriptors obtained by calculating the relative energy contained in various frequency bands. As with the PDF features, the spectral features alone were not sufficient to provide species identification. Another

limitation of the spectral analysis approach is the requirement for a minimum echo length which reduces the number of shoals that could be analyzed using this technique. Moreover the use of a narrowband transducer limits the quality of information that can be obtained using this technique. An image analysis approach similar to the one described in the 1994 paper gave the best classification rate of 57% overall.

Lu and Lee (1995) reported on an echo-signal image processing system (EIPS) developed for fish species identification of fish shoal echograms. Their system measured the following shoal descriptors: (1) area, (2) perimeter, (3) width, (4) height, (5) length, (6) number of pixels, (7) major axis angle, (8) elongation, (9) circularity, (10) rectangularity, (11) mean signal amplitude, (12) standard deviation of amplitude, (13) skewness of signal amplitude, (14) kurtosis of signal amplitude, (15) integrated optical density, (16) horizontal uniformity of optical density , and (17) vertical uniformity of optical density. They used principle component analysis, variable clustering analysis and stepwise discriminant analysis to determine the relationships between the descriptors. The most important descriptors were numbers 1, 2, 3, 4, 6, 7, 9, 10, 11, 12, 13, and 14 as listed above. The accuracy of species identification using the system (with all 17 features) was 98% for the 43 round scad (*Decapterus russelli*) schools, 97% for the 60 anchovy (*Engraulis japonicus*) schools, 94% for the 35 skipjack (*Euthynnus affinis*) schools, 91% for the 42 larval fish schools, and 67% for the 49 horse mackerel (*Decapterus macrosoma*) schools.

Classification has also been attempted using wideband echo-sounders (Magand,

1994, and Zakharia et al., 1996). Magand (1994) describes the use of a 'chirp' echo-sounder to obtain spectral parameters for fish species classification. He describes using auto-regressive (AR) modelling of the echo spectrum from fish shoals and individual fish. Using 30 cepstral coefficients derived from 10 auto-regressive coefficients, a supervised neural network was trained to differentiate between cod (*Gadus morhua*), saithe (*Pollachius virens*) and mackerel (*Scomber scombrus*) in one test and between sardine (*Sardina pilchardus*), anchovy (*Engraulis encrasicholus*), and horse-mackerel (*Trachurus trachurus*) in another test. In a test involving individual caged fish, there was a discrimination rate of 87% (of 133 fish) for mackerel, 72% (of 7 fish) for saithe, and 66% (of 24 fish) for cod. The discrimination rates for fish shoals were as follows: 73% (of 15 shoals) for sardine, 64% (of 10 shoals) for anchovy, and 74% (of 21 shoals) for horse-mackerel.

Zakharia et al. (1996) describe a classification approach based on echo analysis of single pings from a wide-band chirp sounder, operated on a frequency range of 2 octaves (20 kHz to 80 kHz). The classifier used only the spectral signature of the echoes and did not take into account characteristics of school shape. A modeling of the power spectrum of the echo was used to limit the spectral signature to a reduced set of parameters (auto-regressive and cepstral coefficients) that could be used for classification using a neural network. After selecting the echoes corresponding to monospecific catches, only three species remained available for setting up an echo database of single pings, where each ping was used to classify the species. The database consisted of: 270 pings from 15 sardine (*Sardina pilchardus*) schools; 154 pings from 10 schools of anchovy (*Engraulis encrasicolus*); and

465 pings from 21 schools of horse mackerel (*Trachurus turchurus*). The best classification performance (all of the anchovy data was used for training) had an average success rate of 70%. When only 45% of the anchovy data was used for training the average success rate dropped to 54%. While the use of power spectrum information has potential, most fisheries echo-sounders are single frequency and therefore can not supply this information.

Miyashita et al, 1997 obtained multifrequency information by using two single frequency (38 KHz and 120 Khz) transducers simultaneously. By comparing the differences in echo intensities at each frequency they were able to differentiate between isada krill (*Euphausia pacifica*) and walleye pollock (*Theragra chalcogramma*). The authors failed to give specific classification results but this technique seems to have merit although it unfortunately requires the use of two echo-sounder systems.

It is evident by the number of publications on acoustic fish species identification that there are a number of groups working in this area around the world. The author does not know of any groups besides those involved in this research working on automated acoustic identification of species native to Newfoundland coastal waters. A problem with much of the work reported in the literature is a result of the small amount of data available to the researchers. The collection of data for this type of work is very time consuming and expensive (as per a conversation with Dr. George Rose this can range from $500 to $20,000 per day depending on the vessel used). Although many results look impressive it appears that many of these systems have been overtrained, meaning too many features were used

given the small amount of data available. As well, in some of the research reported, classifiers were tested on same data used to train them, a practice which can also give overly optimistic results.

4.0 DATA COLLECTION AND ANALYSIS

The following sections describe the data collection and analysis carried out for this research. Section 4.1 describes the acoustic data and the circumstances under which it was collected in the field. Section 4.2 describes the data pre-processing used before the echogram data was analyzed. Section 4.3 describes the signal and image processing used for segmentation of fish within each echogram. Section 4.4 lists and describes the features extracted for each fish or fish shoal segmented and Section 4.5 describes the classification techniques tested using the extracted features.

4.1 Acoustic Data Collection

The data collected for this study were obtained by Dr. George Rose, the NSERC Chair in Fisheries Conservation at Memorial University of Newfoundland. Dr. Rose collected the data using a BioSonics DT4000 echo-sounder operating with a 38 KHz transducer towed at a depth of approximately 1.5 m below the surface using either a 10 or 30 m vessel. The boat speed was approximately five knots. The transducer beamwidth was 6°, the transmitted pulse length was 0.4 ms and the echo sampling rate was 50 KHz. 50 KHz is a fixed sampling rate for the BioSonics DT4000. Additional transducer parameters can be found in Appendix C.

The capelin data used in this study were collected in Trinity Bay, Newfoundland, in

May 1996. The capelin database was made up of 42 capelin shoals ranging in cross sectional area from 0.9 m^2 to 10,730 m^2 at depths ranging from 53 to 164 m. Figure 8 contains an echogram which contains two capelin shoals.

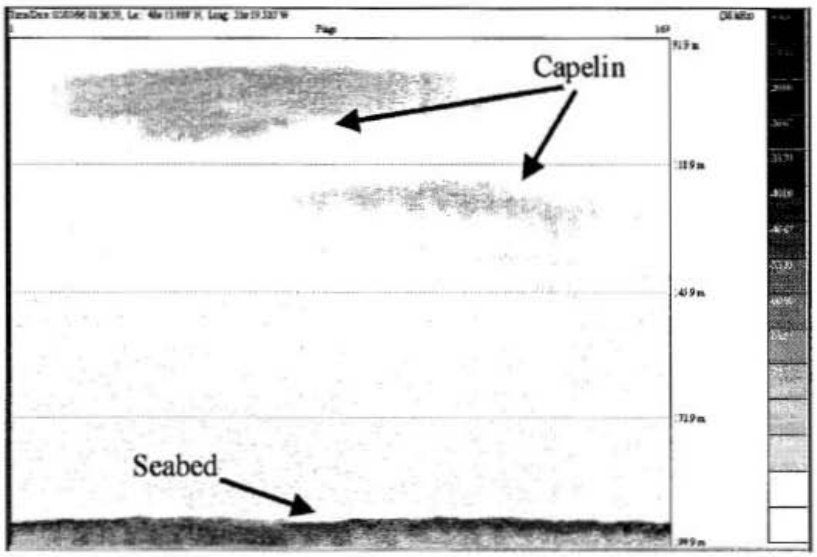


Figure 8: FASIT echogram illustrating distributions of schooling capelin

The cod database consisted of acoustic data from individual fish, small groups of up to approximately ten fish, and cod shoals in which individual fish were difficult to distinguish [see Figures 9 and 10]. The database contained a total of 226 data sets collected in Placentia Bay, on the south coast of Newfoundland, in May, June, and November during the years 1995 to 1997. The smallest individual cod cross sectional area was 0.1 m^2 and the largest shoal was 33,022 m^2. The depth range for the cod data was from 24 to 218 m.

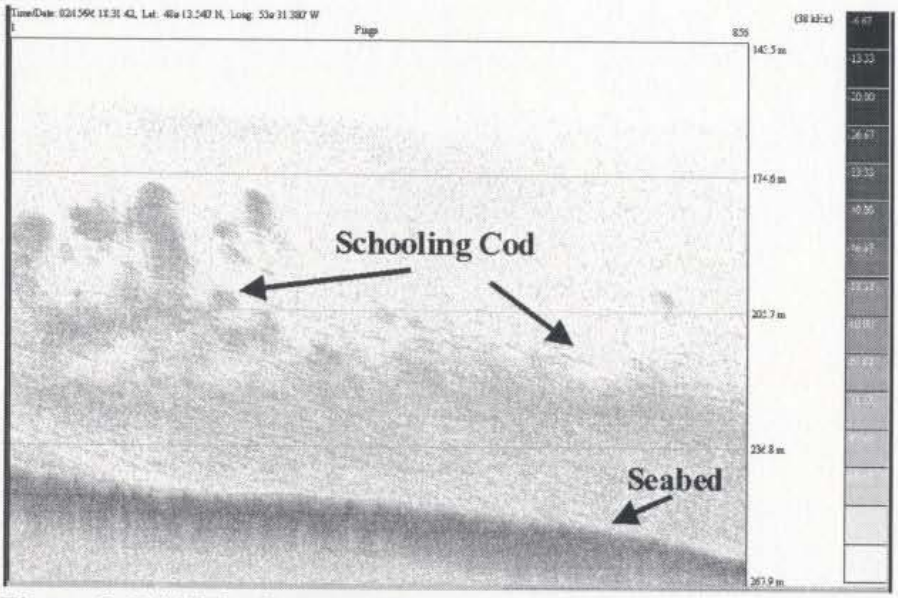


Figure 9: FASIT echogram illustrating distributions of densely schooling cod

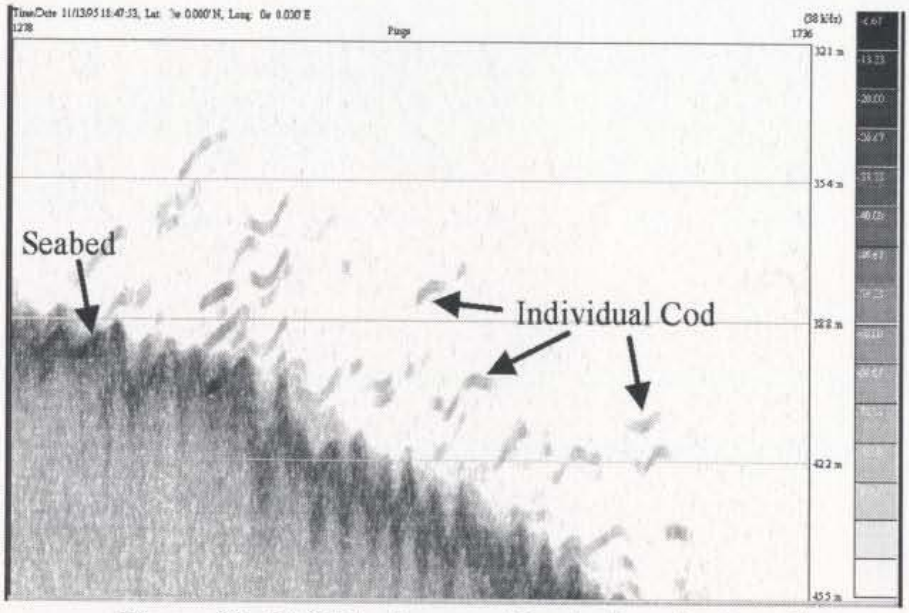


Figure 10: FASIT echogram illustrating single cod

The redfish data used in this study were collected in the 3Ps region of the Grand Bank, off Newfoundland's south coast in June 1996. The redfish data consisted of 134 redfish individuals and small shoals ranging in size from 0.2 m^2 to 144 m^2 and in depth from 106 m to 155 m. Figure 11 is an echogram containing redfish. See Appendix A for maps of the data collection areas for each species.

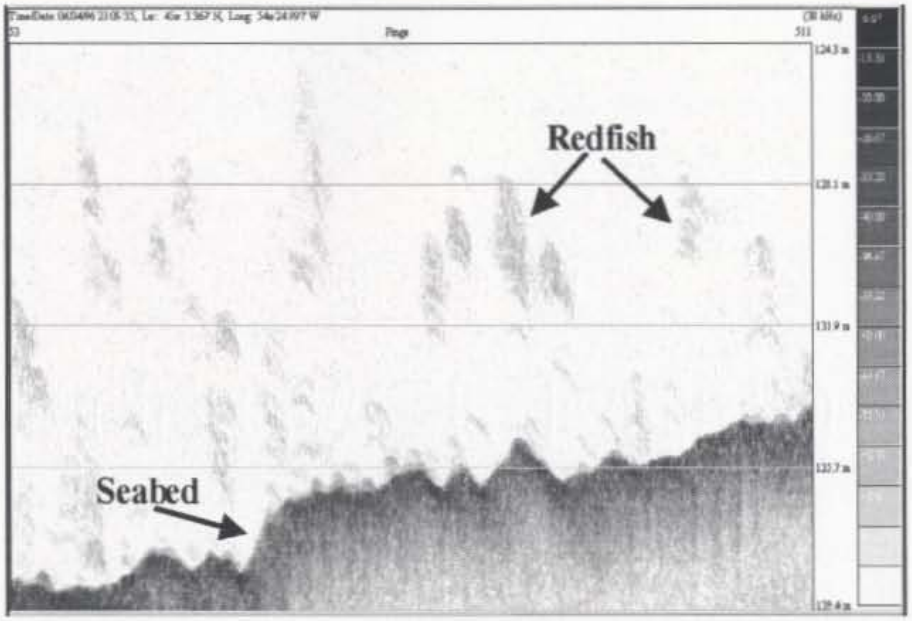


Figure 11: FASIT echogram illustrating single and small groups of redfish

Species composition of the acoustic records was determined by trawling and/or handlining immediately before or after and as close as possible to where the acoustic data collection occurred.

4.2 Data Pre-processing

The acoustic data from each data collection transect was saved as an individual datafile in the BioSonics DT4000 format. Each datafile is read by the FASIT software and used to create an echogram array where the columns of the array represent consecutive pings. The actual physical spacing between pings is estimated using the average boat speed and the ping rate (distance between pings = average boat speed / ping rate). The average boat speed is estimated using GPS information stored within the acoustic data files.

The value of each sample or column element within a ping is determined by the amplitude of the backscattered energy where the amplitude is in units of dB relative to the target strength of a 2 m sphere (this is the measure of amplitude exported by the BioSonics DT4000 echo-sounder). The physical spacing between the rows in the echogram array depends on the sampling rate, and the speed of sound in water (c) (distance between samples = c / sample rate). The value c is calculated based on water temperature and salinity information saved in the header section of each acoustic data file. A typical value for c for the conditions under which data was collected was 1467 m/s. An appropriate absorption coefficient (α) is calculated based on water temperature and salinity and applied to the data along with a 20log(r) time varied gain (TVG), refer to Section 2.1. See Appendix B for the equations used to estimate c and α.

4.3 Segmentation

The first step toward segmentation of fish in the echogram was the detection of the position of the seabed. The bottom returns are usually the samples with the highest amplitudes in a ping. The following bottom detection routine is used. Assume that each ping in a data set contains n sample points. For each sample p, where p ranges from 1 to n-k+1, the window mean amplitude ($\overline{WA}(p)$) is calculated:

$$\overline{WA}(p) = \frac{\sum_{i=p}^{i=p+k-1} amplitude(i)}{k}, \qquad (3)$$

where k is the window height (k = 60 is used). The sample p with the largest window mean amplitude is tagged as the approximate location of the bottom, if the average is greater than -60 dB. Limiting the acceptable returns to those over -60 dB in this manner restricts the data being analyzed to lie in the amplitude range of valid bottom returns. If none of the window mean amplitudes in the ping are above the lower threshold, it is determined that the ping in question does not contain a bottom echo. The above search is performed for each ping (or column) in the echogram array to produce a rough bottom trace. A vertical offset of 1 meter is applied to the bottom trace to move it up to ensure none of the bottom samples remain. To smooth and remove outliers from the offset bottom trace a median filter of size m = 5 is used to replace each bottom point with the median of itself and two points to each side. The samples below the bottom trace are then removed from further analysis by being set to the lowest valid amplitude, -130 dB. The bottom detection algorithm has been summarized in

the block diagram in Figure 12:

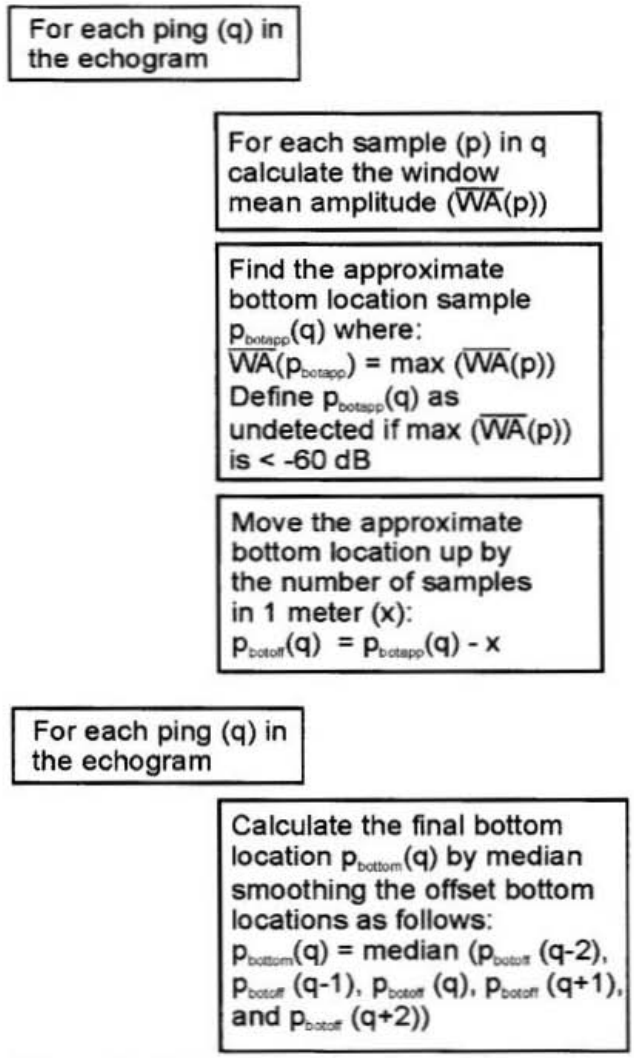


Figure 12: Bottom detection block diagram

After bottom removal, areas containing fish are manually windowed to maximize the number of sample points available when the rectangular widowed region is re-sampled using averaging to fit within a 512 x 512 (or smaller) array where 8-bit resolution is used for each sample amplitude.

In all subsequent operations, the array is treated as an image. The terms sample and pixel will be used interchangeably throughout the remaining discussion. Figure 13 provides an overview all of the image processing steps performed on the echogram images:

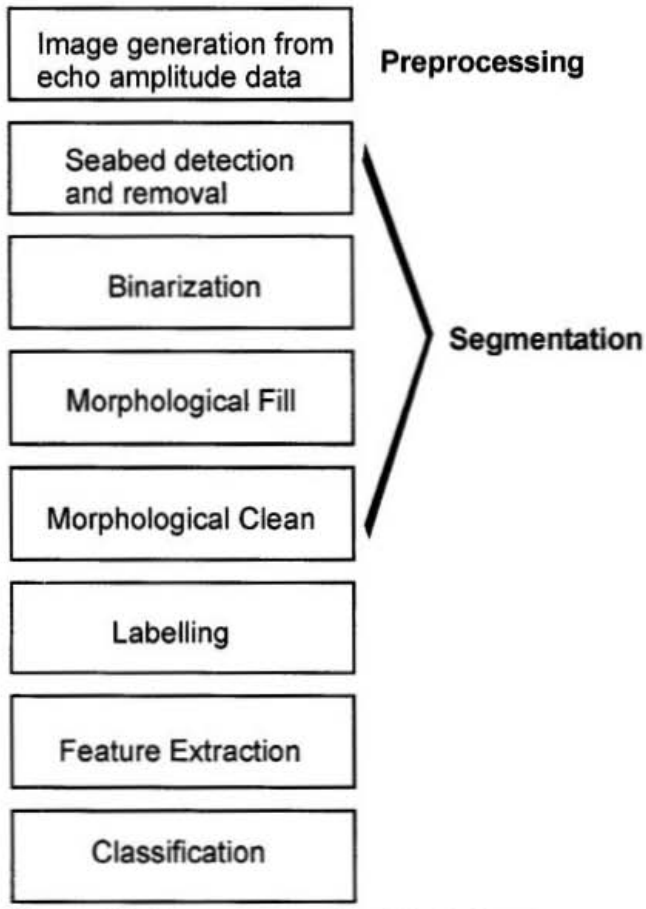


Figure 13: Image processing block diagram

The first image processing function performed on the image is binarization using threshold operation. The thresholding results in the creation of a binary image whose black pixels represent the background and whose white pixels represent the areas containing

acoustic targets where the signal magnitude exceeds the threshold value (-90 dB). Thresholding is followed by a morphological fill and a morphological clean (Gonzalez and Woods, 1993). These operations remove both small objects and small holes inside objects. Figure 14 illustrates the result of the threshold operation and the morphological operations. The objects in the cleaned image are labeled (numbered sequentially) and the features describing each one are extracted as described in the following section. A single object is defined as a collection of white pixels connected to each other vertically, horizontally or diagonally. The image in Figure 14 (c) contains 19 different objects [See Figure 15] . A number of those objects are single fish, evident by their characteristic boomerang shape

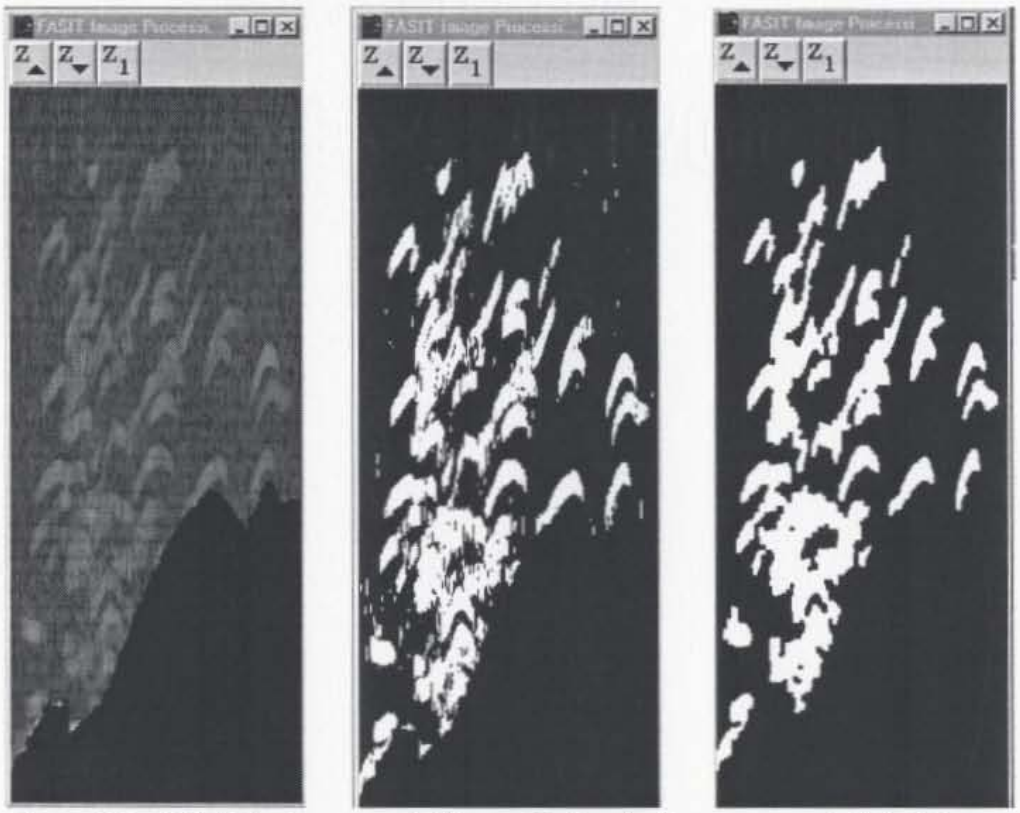


Figure 14: a) Original greyscale "image"after bottom removal, (b) After threshold operation, (c) After morphological operations

while other objects contain multiple fish. It should be noted that each object is classified as a single species. The classifiers for this thesis have been trained and tested on single species objects and not on data were different species are intermingled. The species of interest here rarely intermingle in the waters off Newfoundland therefore this should be adequate for this environment.

Figure 15: Labeled image

4.4 Feature Extraction

The features listed in this section, describing shape, texture, and position are extracted from each object in the segmented image. The Matrox Image Processing Library (MIL) is used to extract or to help derive many of the following features:

1. Area (A): measured in meters ^{2}. The measurement does not include the area of the holes in an object.

2. Perimeter (P): measured in meters.

3. Compactness ($Comp$): a function of area and perimeter, this value is minimum for a circle and increases as shapes become more convoluted.

$$Comp = \frac{P^2}{4\pi A} \tag{4}$$

4. Roughness (R): a measure of the roughness of an object's perimeter. A smooth convex object has the minimum roughness of 1.

$$R = \frac{P}{P_{convex}} \tag{5}$$

where:

P_{convex} = the perimeter of the convex hull of the object in meters (see

Gonzalez and Woods, 1993). The convex hull is approximated using 60 Feret diameters spaced at 3° intervals. A Feret diameter is defined as the distance between parallel tangents touching opposite sides of an object (Russ, 1995).

5. Width (*W*): the width of an object in meters, corrected for echo-sounder beamwidth (Reid and Simmonds, 1993).

$$W = W_{echogram} - 2 \tan\theta \left(\frac{D_{mx} + D_{mn}}{2}\right) \quad \textbf{(6)}$$

where:

$W_{echogram}$ = the measured width of the object in meters,

θ = half angle of the acoustic beam in radians,

D_{mx} = depth, in meters, at the furthest right point of the object,

and D_{mn} = depth, in meters, at the furthest left point of the object.

6. Height (*H*): the distance from the top of an object to the bottom of the object in meters, corrected for pulse length (Reid and Simmonds, 1993):

$$H = H_{echogram} - \frac{\tau}{2} \quad \textbf{(7)}$$

where:

$H_{echogram}$ = the measured height of the object in meters,

and τ = the pulse length in meters

7. Elongation (*El*): a measure of the shape of an object.

$$El = \frac{Length}{Breadth} \tag{8}$$

where:

Length and *Breadth* are calculated by simultaneously solving the following equations for perimeter (*P*) and area (*A*):

$$P = 2\ (Length + Breadth)$$

$$A = Length \times Breadth$$

8. Holes: the number of holes in an object.

9. Axis (*Ax*): the angle at which the maximum diameter is found, it is an indication of the object's orientation in degrees, with positive values indicating a counterclockwise displacement from the "positive X axis." Values can range from 90 to -90 degrees.

10. Mean Amplitude (Amp_{mean}): the average signal amplitude of the samples in the original unthresholded image within the area defined by the object. Amplitude was measured in dB relative to the return from a 2 m diameter sphere.

11. Maximum Amplitude (Amp_{max}): the highest signal amplitude, in dB, of the samples within the area defined by the object.

12. Minimum Amplitude (Amp_{min}): the smallest signal amplitude, in dB, of the samples within the area defined by the object. This value can be less than the lower threshold value (-90 dB) if the morphological fill operation has filled in "holes" initially present in an object.

13. Amplitude Standard Deviation (Amp_{SD}): the amplitude standard deviation, in dB, of the samples within the area defined by the object.

14. Depth to the top of the object (D_{top}): the water depth, in meters, to the top of the object.

15. Depth to the centroid of the object ($D_{centroid}$): the water depth, in meters, to the vertical centroid of the object.

16. Distance from the object to the seabed ($Dist_{bottom}$): the distance, in meters, from the bottom of the object to the seabed. The average seabed depth directly under the object is used to calculate this distance. If there is no seabed present in the echogram, the distance to the bottom boundary of the echogram is used, giving the minimum possible distance to the seabed.

17. Central Moment X0Y2 (*CMY2*): Normalized second order central moment, in dB. A measure of how horizontally dispersed the pixels of an object are from the object's centroid (Glasbey and Horgan, 1995).

$$CMY2 = \frac{\Sigma y_i^2 p_i}{A^2} \quad \textbf{(9)}$$

where:

y_i = horizontal distance from the i'th pixel to the centroid of the object in meters.

p_i = the intensity of the i'th pixel in dB.

18. Central Moment X2Y0 (*CMX2*): Normalized second order central moment, in dB. A measure of how vertically dispersed the pixels of an object are from the object's centroid.

$$CMX2 = \frac{\Sigma x_i^2 p_i}{A^2} \quad \textbf{(10)}$$

where:

x_i = vertical distance from the i'th pixel to the centroid of the object in meters.

p_i = the intensity of the i'th pixel in dB.

19. Central Moment X1Y1 (*CMX1Y1*): Normalized second order central moment, in dB. A measure of how dispersed the pixels of an object are from the object's centroid.

$$CMX1Y1 = \frac{\Sigma x_i y_i p_i}{A^2} \quad \textbf{(11)}$$

where:

x_i = vertical distance from the i'th pixel to the centroid of the object in meters.

y_i = horizontal distance from the i'th pixel to the centroid of the object in meters.

p_i = the intensity of the i'th pixel in dB.

20. Binarized Central Moment X0Y2 (*BCMY2*): Normalized binary second order central moment. A measure of how horizontally dispersed the pixels of an object are from the object's centroid (Glasbey and Horgan, 1995).

$$BCMY2 = \frac{\Sigma y_i^2}{A^2} \quad \textbf{(12)}$$

where:

y_i = horizontal distance from the i'th pixel to the centroid of the object in meters.

21. Binarized Central Moment X2Y0 (*BCMX2*): Normalized binary second order central moment. A measure of how vertically dispersed the pixels in an object are from the object's centroid.

$$BCMX2 = \frac{\Sigma x_i^2}{A^2} \quad \textbf{(13)}$$

where:

x_i = vertical distance from the i'th pixel to the centroid of the object in meters.

22. Binarized Central Moment X1Y1 (*BCMX1Y1*): Normalized binary second order central moment. A measure of how dispersed the pixels of an object are from the object's centroid.

$$BCMX1Y1 = \frac{\Sigma x_i y_i}{A^2} \quad \textbf{(14)}$$

where:

x_i = vertical distance from the i'th pixel to the centroid of the object in meters.

y_i = horizontal distance from the i'th pixel to the centroid of the object in meters.

4.5 Classifier Design

Training sets of 226 cod, 134 redfish, and 42 capelin objects (as object has been defined in Section 4.5) have been used to generate a database of features for each species. Using these data a number of different classifier designs, based on 3 nearest-neighbor and Mahalanobis distance, have been built and tested. The following sections describe the steps taken for feature reduction, and the types of classifiers assessed.

4.5.1 Feature Reduction

The first step in any classifier design is the analysis of the available feature data. This analysis results in the removal of features which are biased or noisy and the removal or combination of highly correlated features. The number of features used by the final classifier should be as low as possible. A large number of features can lead to overtraining, meaning that the classifier is too specialized for the training data, which can cause the classifier to perform poorly on unseen data. A reduction in the number of features that ultimately need to be calculated will also improve the speed performance of the software.

After the removal of biased features, two types of algorithms were used for further feature reduction: a "filter" technique which evaluated the data based on a measure of redundancy and a "wrapper" technique which evaluated the usefulness of each feature given a specific classifier design (Hall, 2000). Section 4.5.1.1 will describe biased feature removal, Section 4.5.1.2 will describe the filter algorithm used and Section 4.5.1.3 will describe the wrapper algorithms used.

4.5.1.1 Biased Feature Removal

Features describing shoal depth (D_{top} and $D_{centroid}$) were removed from the feature list because they were biased. The acoustic data available was from a limited number of surveys in a limited number of regions; therefore, including these features would give falsely high classification results. Figure 16 presents the depth data for each database.

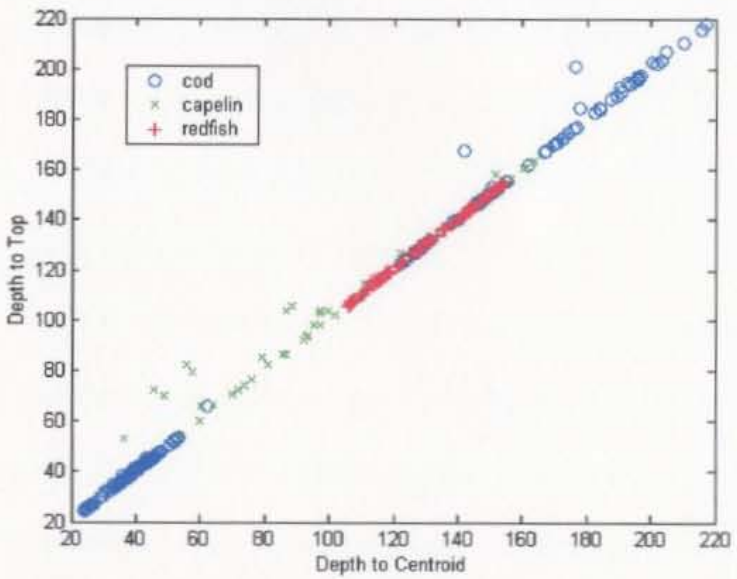


Figure 16: Feature plot of $D_{centroid}$ and D_{top} (in meters)

4.5.1.2 Redundant Feature Removal

To reduce the number of remaining features, factor analysis was used. Factor analysis reduces the complexity of a classifier by combining or removing features which are highly correlated (Duda and Hart, 1973). The correlation between two features can be defined as:

$$\rho_{ij} = \frac{\sigma_{ij}}{\sqrt{\sigma_{ii}\,\sigma_{jj}}} \qquad (15)$$

where σ_{ij} is the covariance of the features i and j; σ_{ii} and σ_{jj} are the variances of the features i and j, respectively. Completely uncorrelated features will have $\rho_{ij} = 0$, and completely correlated features will have $\rho_{ij} = 1$. If the correlation between two features was above a specified threshold for all three species (a threshold of 0.9 was used in this case) those features were considered for removal. During the feature reduction procedure, one feature from each pair of highly correlated features was temporarily removed from the database and a Mahalanobis classifier was designed with the remaining features. If classification performance was not degraded or if performance was improved, one of those features was permanently removed from the database. Using this technique, it was possible to remove the following three features : *Elongation* (closely correlated with *Compactness*), *BCMY2* (closely correlated with *CMY2*), *and CMX2* (closely correlated with *BCMX2*). Correlation is evident in Figures 17 and 18 illustrating *Elongation* vs. *Compactness* and *CMY2* vs *BCMY2* respectively. Redundant feature removal as well as the removal of the biased depth features

left 17 features that could be used by the classifiers tested.

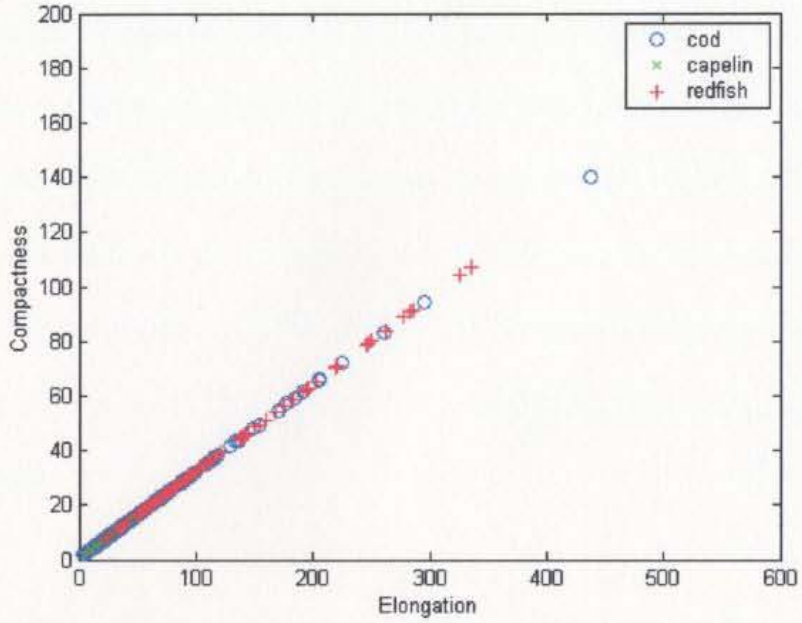


Figure 17: Feature plot showing *El* and *Comp*

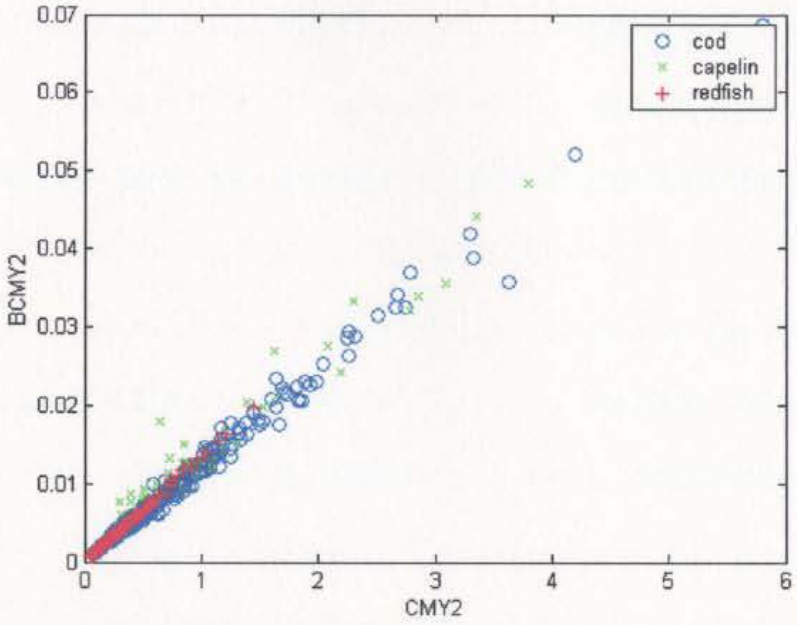


Figure 18: Feature plot showing CMY2 and BCMY2

4.5.1.3 Classifier Specific Feature Reduction

Trying every combination of the remaining 17 features to find the optimum feature set for the different types of classifiers to be tested was impractical. Two well known "wrapper" feature reduction techniques, sequential backward selection (SBS) and sequential forward selection (SFS) were used to reduce the number of features for each classifier developed. The chosen feature list for a given classifier was the best one found using both SBS and SFS [see Appendix D for Matlab© code]. Please note that the Matlab © code for feature reduction was written by the author, as was all of the Matlab © and C code referred to in this thesis.

Sequential backward selection starts with all of the available features. One feature is removed at a time and the resulting classifier is tested. The feature whose removal causes the most improvement in classifier performance is permanently removed. This cycle is repeated until all but one of the features remains. For this study the favored feature list was the one with 10 or fewer features which gave the best classification performance. An upper limit of 10 features was chosen somewhat arbitrarily but based on the author's previous experience with pattern recognition problems of a similar nature. This number was shown to be appropriate when leave-one-out testing was performed on the various classifiers. Performance usually dropped or stayed the same when more than 10 features were used, indicating that the classifiers with more than 10 features were probably overtrained. Figures 19 and 20 show the average classification performance of two single node classifiers (3-NN and Mahalanobis respectively) , where performance was determined using leave-one-out

testing. It can be seen that performance was not improved by the inclusion of more than 10 features in either of these cases.

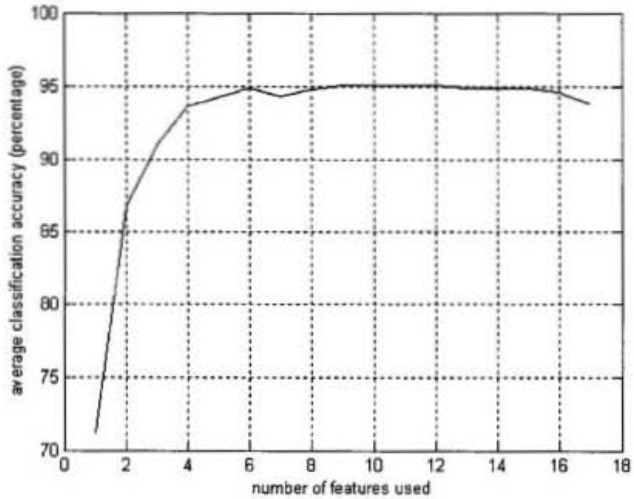


Figure 19: Single node 3NN classifier accuracy as a function of number of features used

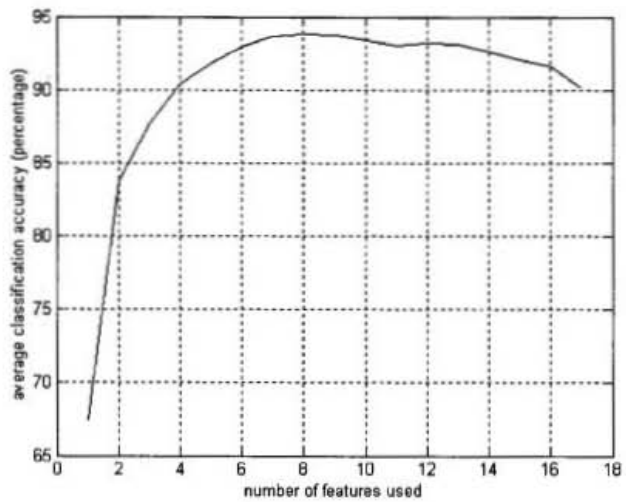


Figure 20: Single node Mahalanobis classifier accuracy as a function of number of features used

Sequential forward selection starts with no features. One feature at a time is added to the available feature list and the resulting classifier is tested. The feature whose inclusion causes the most improvement in classifier performance is permanently included. For this study this cycle was repeated until 10 features were included. As with SBS the favored feature list was the one with 10 or fewer features which gave the best classification performance.

4.5.2 Classifier Types

Two different classifier types were implemented and tested using: a 3-Nearest Neighbor classifier and a Mahalanobis distance classifier. These classifiers are described in the following two sections.

4.5.2.1 Three Nearest-Neighbor Classifier

A nearest neighbor classifier is essentially a look-up table. It is completely non-parametric, meaning it assumes nothing about the population and makes no generalizations about the population (Weiss and Kulikowski, 1991). With the three nearest-neighbor (3-NN) classifiers that were implemented and tested, features were normalized and the Euclidean distance was used. An unknown data point was said to be of the species that showed up at least twice out of the three nearest neighbors. Since there were three possible species classes a tie could result. In this situation the single nearest neighbor was used.

4.5.2.2 Mahalanobis Distance Classifier

Often a distance measure known as the Mahalanobis distance is useful for classification. This parametric classifier measures the distance in feature space from an unknown object, x, to the mean of class i using the following formula:

$$Dist_i = (x - m_i)^T K_i^{-1} (x - m_i) \quad \textbf{(16)}$$

where m_i is the class (species in our case) mean and K_i is the class covariance matrix calculated from the training data . The use of the class covariance matrix in the distance measure takes into account the correlation among the features and ensures that the distance measure is unaffected by scale changes between features. It also ensures that the variance in cluster "shapes" is taken into account.

The mean and class covariance matrices were computed for each species (Table 4, contains the mean feature values for cod, capelin and redfish). The distance from the unknown object, x, and each class was computed using the above equation. The unknown object was classified as the class corresponding to the minimum distance. This classification technique is quite simple, easy to implement, and, unlike many distance classifiers, the recognition speeds are very reasonable because it is unnecessary to compute the distances to all objects in each class.

Table 4: Mean feature values for each species

Feature	Mean Values		
	Cod	Capelin	Redfish
Area	188.097 m^2	946.2026 m^2	15.7034 m^2
Perimeter	76.86 m	169.03 m	74.01 m
Compactness	16.38	8.31	34.9
Roughness	5.15	4.34	10.2
Width	15.60 m	41.77 m	12.64 m
Height	1.31 m	9.77 m	0.91 m
Elongation	49.41	24.05	107.64
Holes	0.45	0.76	0.18
Axis	2.7 °	5.1°	-1.5°
Mean Amplitude	-83.71 dB	-88.84 dB	-87.56 dB
Maximum Amplitude	-74.53 dB	-73.58 dB	-79.68 dB
Minimum Amplitude	-101.79 dB	-122.14 dB	-105.54 dB
Amplitude Standard Deviation	5.33 dB	10.91 dB	4.12 dB
Depth to the top of the object	77.84 m	97.56 m	131.98 m
Depth to the centroid of the object	78.51 m	102.52 m	132.15 m
Distance from the object to the seabed	14.83 m	91.47 m	2.23 m
Central Moment X0Y2	0.92 dB	1.18 dB	0.34 dB
Central Moment X2Y0	190.24 dB	52.07 dB	240.94 dB
Central Moment X1Y1	-1.424 dB	-0.83 dB	1.86 dB
Binarized Central Moment X0Y2	0.011	0.016	0.005
Binarized Central Moment X2Y0	2.366	0.742	3.263
Binarized Central Moment X1Y1	-0.017	-0.01	0.025

4.5.3 Classifier Configurations

Two classifier "tree structures" were tested for each classifier type (3-NN and Mahalanobis Distance). These configurations are illustrated in Figures 21 and 22. The

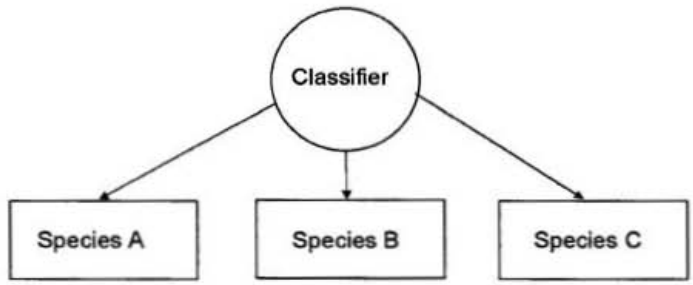


Figure 21: Single-node classifier

configuration illustrated in Figure 21 is a single node classifier with three possible outputs: species A, B or C based on a common set of features. The configuration in Figure 22 contains two nodes (or classifiers) which use two different sets of features. Classifier 1 will classify an object as being species A or "not species A". If an object is classified as "not

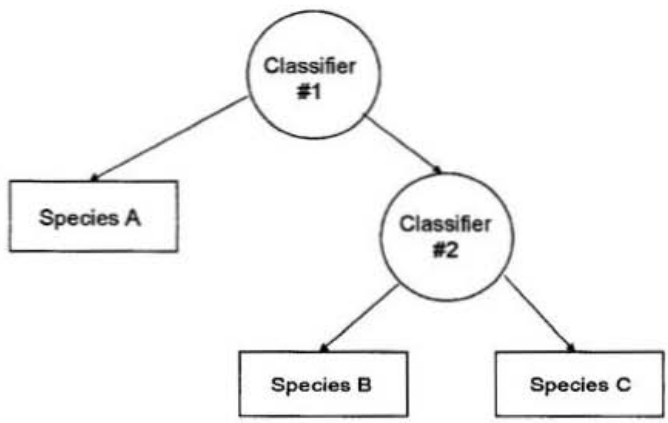


Figure 22: Dual node classifier tree structure

Species A" classifier #2 will classify the object as Species B or Species C. This configuration is useful if it is possible to remove one class from the others with very few features or if there is a limited amount of data for one class which forces the use of a limited number of features for its detection. Table 5 lists the fourteen classifiers combinations of types and configurations implemented and tested.

Table 5: Classifier combinations

Config. #	Node 1 Classifier	Node 2 Classifier	Species A	Species B	Species C
1	3 NN	NA	cod	capelin	redfish
2	3 NN	3 NN	cod	capelin	redfish
3	3 NN	3 NN	capelin	cod	redfish
4	3 NN	3 NN	redfish	cod	capelin
5	Mah.Dist.	NA	cod	capelin	redfish
6	Mah.Dist.	Mah.Dist.	cod	capelin	redfish
7	Mah.Dist.	Mah.Dist.	capelin	cod	redfish
8	Mah.Dist.	Mah.Dist.	redfish	cod	capelin
9	3 NN	Mah.Dist	cod	capelin	redfish
10	3 NN	Mah.Dist	capelin	cod	redfish
11	3 NN	Mah.Dist	redfish	cod	capelin
12	Mah.Dist	3 NN	cod	capelin	redfish
13	Mah.Dist	3 NN	capelin	cod	redfish
14	Mah.Dist	3 NN	redfish	cod	capelin

5.0 RESULTS AND DISCUSSION

Section 5.1 will provide the classification results for each classifier design listed in Table 6. The results have been given for "leave-one-out" tests. The leave-one-out test is a simple technique for estimating classifier error rates. For a given sample size x, a classifier is generated using (x-1) cases and tested on the single remaining case. This is repeated x times, each time regenerating the classifier by leaving one sample out. By doing this, each sample is used as a test case thus maximizing the number of tests performed on "unseen" data. This technique provides an almost unbiased estimation of a classifier's error rate (Weisse and Kulikowski, 1991). It is however very time consuming and only practical to use when sample sizes are small, as they were with this research and when classifier construction is not time consuming.

5.1 Classifier Performance

5.1.1 Three-Nearest Neighbor Classifiers

5.1.1.1 Classifier Configuration #1

Classifier Configuration #1 is a single node 3-NN classifier. The feature set used by this classifier was found with backward feature selection and includes the following nine features: *Mean Amplitude, Maximum Amplitude, Minimum Amplitude, Amplitude Standard Deviation, Distance from Object to Seabed, Central Moment X0Y2, Central Moment X1Y1, Binarized Central Moment X2Y0*, and *Binarized Central Moment X1Y1*. The confusion

matrix for this classifier is presented in Table 6.

Table 6: Confusion matrix for classifier configuration #1

True Class	Predicted Class Membership		
	Cod	**Capelin**	**Redfish**
Cod	208 (92.0%)	2 (0.9%)	16 (7.1%)
Capelin	2 (4.8%)	40 (95.2%)	0
Redfish	3 (2.2%)	0	131 (97.8%)

Please note that the percentages given in the confusion matrix are the percentage of the true class species database classified as the predicted class species.

5.1.1.2 Classifier Configuration #2

Classifier Configuration #2 is a dual node classifier where both nodes are 3-NN classifiers. The feature set used by the first node (for the identification of cod) was found using backward feature selection and includes the following ten features: *Compactness, Roughness, Height, Mean Amplitude, Minimum Amplitude, Amplitude Standard Deviation, Distance from Object to Seabed, Central Moment X1Y1, Binarized Central Moment X2Y0,* and *Binarized Central Moment X1Y1.* The feature set used by the second node (for the identification of capelin and redfish) was found using backward feature selection and includes the following seven features: *Compactness, Mean Amplitude, Minimum Amplitude, Distance from Object to Seabed, Central Moment X1Y1, Binarized Central Moment X2Y0,* and *Binarized Central Moment X1Y1.* The confusion matrix for this classifier is presented

in Table 7.

Table 7: Confusion matrix for classifier configuration #2

True Class	Predicted Class Membership		
	Cod	**Capelin**	**Redfish**
Cod	204 (90.3%)	2 (0.9%)	20 (8.8%)
Capelin	1 (2.4%)	40 (95.2%)	1 (2.4%)
Redfish	2 (1.5%)	0	132 (98.5%)

5.1.1.3 Classifier Configuration #3

Classifier Configuration #3 is a dual node classifier where both nodes are 3-NN classifiers. The feature set used by the first node (for the identification of capelin) was found using backward feature selection and includes the following six features: *Compactness, Minimum Amplitude, Distance from Object to Seabed, Central Moment X1Y1, Binarized Central Moment X2Y0,* and *Binarized Central Moment X1Y1.* The feature set used by the second node (for the identification of cod and redfish) was found using backward feature selection and includes the following five features: *Roughness, Mean Amplitude, Amplitude Standard Deviation, Distance from Object to Seabed,* and *Central Moment X0Y2.* The confusion matrix for this classifier is presented in Table 8.

Table 8: Confusion matrix for classifier configuration #3

True Class	Predicted Class Membership		
	Cod	**Capelin**	**Redfish**
Cod	213 (94.2%)	1 (0.4%)	12 (5.3%)
Capelin	1 (2.4%)	41 (97.6%)	0
Redfish	2 (1.5%)	0	132 (98.5%)

5.1.1.4 Classifier Configuration #4

Classifier Configuration #4 is a dual node classifier where both nodes are 3-NN classifiers. The feature set used by the first node (for the identification of redfish) was found using backward feature selection and includes the following ten features: *Roughness, Mean Amplitude, Maximum Amplitude, Minimum Amplitude, Amplitude Standard Deviation, Distance from Object to Seabed, Holes, Central Moment X0Y2, Binarized Central Moment X2Y0*, and *Binarized Central Moment X1Y1*. The feature set used by the second node (for the identification of capelin and cod) was found using forward feature selection and includes the following eight features: *Compactness, Roughness, Height, Axis, Mean Amplitude, Amplitude Standard Deviation, Distance from Object to Seabed*, and *Central Moment X0Y2*. The confusion matrix for this classifier is presented in Table 9.

Table 9: Confusion matrix for classifier configuration #4

True Class	Predicted Class Membership		
	Cod	**Capelin**	**Redfish**

Cod	211 (93.4%)	0	15 (6.6%)
Capelin	2 (4.8%)	40 (95.2%)	0
Redfish	2 (1.5%)	0	132 (98.5%)

5.1.2 Mahalanobis Distance Classifiers

5.1.2.1 Classifier Configuration #5

Classifier Configuration #5 is a single node Mahalanobis Distance classifier. The feature set used by this classifier was found with backward feature selection and includes the following eight features: *Perimeter, Compactness, Roughness, Height, Axis, Minimum Amplitude, Amplitude Standard Deviation*, and *Distance from Object to Seabed.* The confusion matrix for this classifier is presented in Table 10.

Table 10: Confusion matrix for classifier configuration #5

	Predicted Class Membership		
True Class	**Cod**	**Capelin**	**Redfish**
Cod	214 (94.7%)	4 (1.8%)	8 (3.5%)
Capelin	1 (2.4%)	41 (97.6%)	0
Redfish	23 (17.2%)	0	111 (82.8%)

5.1.2.2 Classifier Configuration #6

Classifier Configuration #6 is a dual node classifier where both nodes are Mahalanobis Distance classifiers. The feature set used by the first node (for the

identification of cod) was found by backward feature selection and includes the following eight features: *Perimeter, Compactness, Roughness, Height, Axis, Minimum Amplitude, Amplitude Standard Deviation*, and *Distance from Object to Seabed.* The feature used by the second node (for the identification of capelin and redfish) was found using forward feature selection and includes the following four features: *Roughness, Axis, Maximum Amplitude*, and *Distance from Object to Seabed.* The confusion matrix for this classifier is presented in Table 11.

Table 11: Confusion matrix for classifier configuration #6

True Class	Predicted Class Membership		
	Cod	**Capelin**	**Redfish**
Cod	214 (94.7%)	2 (0.9%)	10 (4.4%)
Capelin	1 (2.4%)	41 (97.6%)	0
Redfish	23 (17.2%)	1 (0.7%)	110 (82.1%)

5.1.2.3 Classifier Configuration #7

Classifier Configuration #7 is a dual node classifier where both nodes are Mahalanobis distance classifiers. The feature set used by the first node (for the identification of capelin) was found using backward feature selection and includes the following six features: *Compactness, Roughness, Axis, Minimum Amplitude, Distance from Object to Seabed*, and *Holes*. The feature set used by the second node (for the identification of cod and redfish) was found using backward feature selection and includes the following eight

features: *Perimeter, Compactness, Roughness, Height, Axis, Minimum Amplitude, Amplitude Standard Deviation*, and *Distance from Object to Seabed.* The confusion matrix for this classifier is presented in Table 12.

Table 12: Confusion matrix for classifier configuration #7

True Class	Predicted Class Membership		
	Cod	**Capelin**	**Redfish**
Cod	217 (96.0%)	1 (0.4%)	8 (3.5%)
Capelin	7 (7.1%)	39 (92.9%)	0
Redfish	23 (17.2%)	0	111 (82.8%)

5.1.2.4 Classifier Configuration #8

Classifier Configuration #8 is a dual node classifier where both nodes are Mahalanobis distance classifiers. The feature set used by the first node (for the identification of redfish) was found using backward feature selection and includes the following nine features: *Perimeter, Compactness, Roughness, Height, Axis, Minimum Amplitude, Amplitude Standard Deviation, Distance from Object to Seabed*, and *Holes.* The feature set used by the second node (for the identification of capelin and cod) was found using forward feature selection and includes the following ten features: *Perimeter, Compactness, Height, Minimum Amplitude, Amplitude Standard Deviation, Distance from Object to Seabed, Holes, Central Moment X0Y2, Central Moment X1Y1*, and *Binarized Central Moment X1Y1.* The confusion matrix for this classifier is presented in Table 13.

Table 13: Confusion matrix for classifier configuration #8

True Class	Predicted Class Membership		
	Cod	**Capelin**	**Redfish**
Cod	211 (93.4%)	5 (2.2%)	10 (4.4%)
Capelin	3 (7.1%)	39 (92.9%)	0
Redfish	23 (17.2%)	0	111 (82.8%)

5.1.3 Combination Classifiers

5.1.3.1 Classifier Configuration #9

Classifier Configuration #9 is a dual node classifier where the first node is a 3NN classifier and the second node is a Mahalanobis classifier. The feature set used by the first node (for the identification of cod) was found using backward feature selection and includes the following ten features: *Compactness, Roughness, Height, Mean Amplitude, Minimum Amplitude, Amplitude Standard Deviation, Distance from Object to Seabed, Central Moment X1Y1, Binarized Central Moment X2Y0*, and *Binarized Central Moment X1Y1*. The feature set used by the second node (for the identification of capelin and redfish) was found using forward feature selection and includes the following four features: *Roughness, Axis, Maximum Amplitude*, and *Distance from Object to Seabed.* The confusion matrix for this classifier is presented in Table 14.

Table 14: Confusion matrix for classifier configuration #9

True Class	Predicted Class Membership		
	Cod	**Capelin**	**Redfish**
Cod	204 (90.3%)	8 (3.5%)	14 (6.2%)
Capelin	1 (2.4%)	41 (97.6%)	0
Redfish	2 (1.5%)	1 (0.7%)	131 (97.8%)

5.1.3.2 Classifier Configuration #10

Classifier Configuration #10 is a dual node classifier where the first node is a 3NN classifier and the second node is a Mahalanobis classifier. The feature set used by the first node (for the identification of capelin) was found using backward feature selection and uses the following six features: *Compactness, Minimum Amplitude, Distance from Object to Seabed, Central Moment X1Y1, Binarized Central Moment X2Y0*, and *Binarized Central Moment X1Y1.* The feature set used by the second node (for the identification of cod and redfish) was found using backward feature selection and includes the following eight features: *Perimeter, Compactness, Roughness, Height, Axis, Minimum Amplitude, Amplitude Standard Deviation*, and *Distance from Object to Seabed.* The confusion matrix for this classifier is presented in Table 15.

Table 15: Confusion matrix for classifier configuration #10

True Class	Predicted Class Membership		
	Cod	**Capelin**	**Redfish**

Cod	217 (96.0%)	1 (0.4%)	8 (3.5%)
Capelin	1 (2.4%)	41 (97.6%)	0
Redfish	23 (17.2%)	0	111 (82.8%)

5.1.3.3 Classifier Configuration #11

Classifier Configuration #11 is a dual node classifier where the first node is a 3NN classifier and the second node is a Mahalanobis classifier. The feature set used by the first node (for the identification of redfish) was found using backward feature selection and uses the following ten features: *Roughness, Mean Amplitude, Maximum Amplitude, Minimum Amplitude, Amplitude Standard Deviation, Distance from Object to Seabed, Holes, Central Moment X0Y2, Binarized Central Moment X2Y0*, and *Binarized Central Moment X1Y1*. The feature set used by the second node (for the identification of cod and capelin) was found using forward feature selection and includes the following ten features: *Perimeter, Compactness, Height, Minimum Amplitude, Amplitude Standard Deviation, Distance from Object to Seabed, Holes Central Moment X0Y2, Central Moment X1Y1*, and *Binarized Central Moment X1Y1*. The confusion matrix for this classifier is presented in Table 16.

Table 16: Confusion matrix for classifier configuration #11

	Predicted Class Membership		
True Class	**Cod**	**Capelin**	**Redfish**
Cod	206 (91.2%)	5 (2.2%)	15 (6.6%)
Capelin	3 (7.1%)	39 (92.9%)	0

Redfish	2 (1.5%)	0	132 (98.5%)

5.1.3.4 Classifier Configuration #12

Classifier Configuration #12 is a dual node classifier where the first node is a Mahalanobis classifier and the second node is a 3NN classifier. The feature set used by the first node (for the identification of cod) was found using backward feature selection and uses the following eight features: *Perimeter, Compactness, Roughness, Height, Axis, Minimum Amplitude, Amplitude Standard Deviation*, and *Distance from Object to Seabed.* The feature set used by the second node (for the identification of capelin and redfish) was found using backward feature selection and includes the following seven features: *Compactness, Mean Amplitude, Minimum Amplitude, Distance from Object to Seabed, Central Moment X1Y1, Binarized Central Moment X2Y0*, and *Binarized Central Moment X1Y1.* The confusion matrix for this classifier is presented in Table 17.

Table 17: Confusion matrix for classifier configuration #12

	Predicted Class Membership		
True Class	**Cod**	**Capelin**	**Redfish**
Cod	214 (94.7%)	1 (0.4%)	11 (4.9%)
Capelin	1 (2.4%)	41 (97.6%)	0
Redfish	23 (17.2%)	0	111 (82.8%)

5.1.3.5 Classifier Configuration #13

Classifier Configuration #13 is a dual node classifier where the first node is a Mahalanobis classifier and the second node is a 3NN classifier. The feature set used by the first node (for the identification of capelin) was found using backward feature selection and uses the following six features: *Compactness, Roughness, Axis, Minimum Amplitude, Distance from Object to Seabed,* and *Holes*. The feature set used by the second node (for the identification of cod and redfish) was found using backward feature selection and includes the following five features: *Roughness, Mean Amplitude, Amplitude Standard Deviation, Distance from Object to Seabed,* and *Central Moment X0Y2.* The confusion matrix for this classifier is presented in Table 18.

Table 18: Confusion matrix for classifier configuration #13

True Class	Predicted Class Membership		
	Cod	**Capelin**	**Redfish**
Cod	214 (94.7%)	1 (0.4%)	11 (4.9%)
Capelin	3 (7.1%)	39 (92.9%)	0
Redfish	2 (1.5%)	0	132 (98.5%)

5.1.3.6 Classifier Configuration #14

Classifier Configuration #14 is a dual node classifier where the first node is a Mahalanobis classifier and the second node is a 3NN classifier. The feature set used by the first node (for the identification of redfish) was found using backward feature selection and

uses the following nine features: *Perimeter, Compactness, Roughness, Height, Axis, Minimum Amplitude, Amplitude Standard Deviation, Distance from Object to Seabed*, and *Holes*. The feature set used by the second node (for the identification of cod and capelin) was found using backward feature selection and includes the following eight features: *Compactness, Roughness, Height, Axis, Mean Amplitude, Amplitude Standard Deviation, Distance from Object to Seabed*, and *Central Moment X0Y2*. The confusion matrix for this classifier is presented in Table 19.

Table 19: Confusion matrix for classifier configuration #14

	Predicted Class Membership		
True Class	**Cod**	**Capelin**	**Redfish**
Cod	216 (95.6%)	0	10 (4.4%)
Capelin	2 (4.8%)	40 (95.2%)	0
Redfish	23 (17.2%)	0	111 (82.8%)

5.2 Discussion

The best average classification accuracy (96.8%) was produced by classifier #3, the dual node classifier where both nodes were 3-NN classifiers and the first node was used to identify capelin. Based on average classification accuracy, the 3-NN classifiers outperformed the Mahalanobis distance classifiers by about 5.0% (as shown in Table 20). The

performance of the combination classifiers were in general between the performance of the 3-NN classifiers and the Mahalanobis distance classifiers. The Mahalanobis distance classifiers were relatively poor at separating redfish from cod and capelin but very good at separating cod from the other two species. The 3-NN classifiers had a more consistent performance for each of the three species'.

There was no significant difference in the number of features required for the different classifier configurations. It is interesting to note that the two classifiers with the best performance also used the fewest number of features per node. The *Distance from the Object to the Seabed* feature was used by every node in every classifier, as was at least one amplitude feature. As expected, amplitude features were very useful but never sufficient for classification. The Mahalanobis distance classifiers made use of morphological shape and size features such as *Area, Compactness, Width* and *Height* more often than the 3-NN classifiers [see Figure 23]. The 3-NN classifiers used the binarized central moments which are shape descriptors more often than the Mahalanobis distance classifiers [see Figure 24]. For the 3-NN classifiers, the amplitude features were very important [see Figure 25].

Although it did not give the best classification result, for speed optimization it was decided that a Mahalanobis distance based classifier would be implemented in the FASIT software (it was anticipated that a great deal of training data would be available to the software in the future and a nearest neighbour classifier would be too slow). Further, since there was limited capelin data (42 feature sets) the classifier chosen (to avoid overtraining),

was the one that used the smallest number of features for the classification of capelin - classifier configuration #7. Given the speed of computers today, classification speed is no longer an issue for this application. I would therefore recommend that the classifier configuration that gave the best overall results (configuration #3) be implemented in newer versions of FASIT. This classifier not only provided the best classification accuracy but it required on average the lowest number of features.

Table 20: Summary of classifier results

Class #	Node 1		Node 2		Classification Accuracy			Average
	type	num	type	num	Cod	Capelin	Redfish	Accuracy
#1	3NN	9	NA	NA	92	95.2	97.8	95
#2	3NN	10	3NN	7	90.3	95.2	98.5	94.7
#3	3NN	6	3NN	5	94.2	97.6	98.5	96.8
#4	3NN	10	3NN	8	93.4	95.2	98.5	95.7
#5	Mah.	8	NA	NA	94.7	97.6	82.8	91.7
#6	Mah.	8	Mah.	4	94.7	97.6	82.1	91.5
#7	Mah.	6	Mah.	8	96	92.9	82.8	90.6
#8	Mah.	9	Mah.	10	93.4	92.9	82.8	89.7
#9	3NN	10	Mah	4	90.3	97.6	97.8	95.2
#10	3NN	6	Mah	8	96	97.6	82.8	92.1
#11	3NN	10	Mah	10	91.2	92.9	98.5	94.2
#12	Mah	8	3NN	7	94.7	97.6	82.8	91.7
#13	Mah	6	3NN	5	94.7	92.9	98.5	95.4
#14	Mah	9	3NN	8	95.6	95.2	82.8	91.2

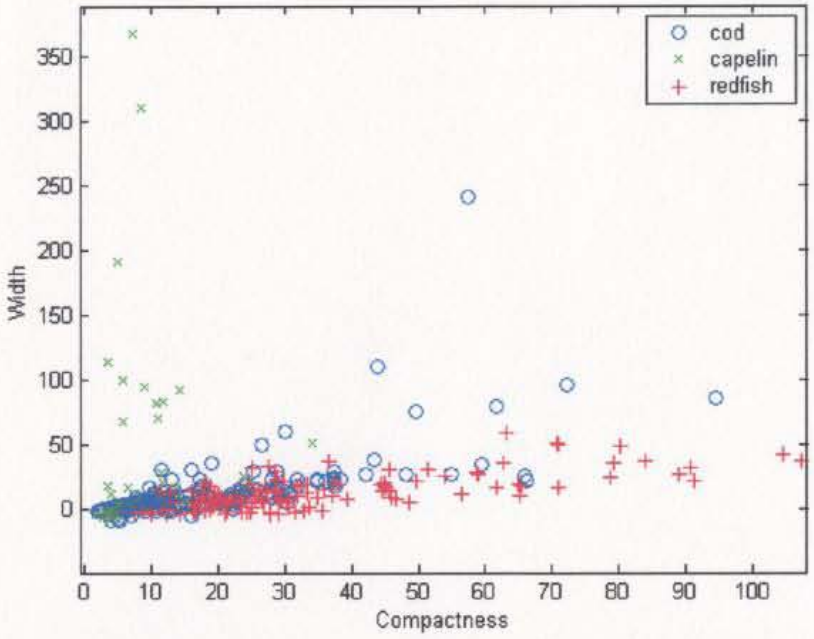


Figure 23: Feature plot for *Comp* and *W (meters)*

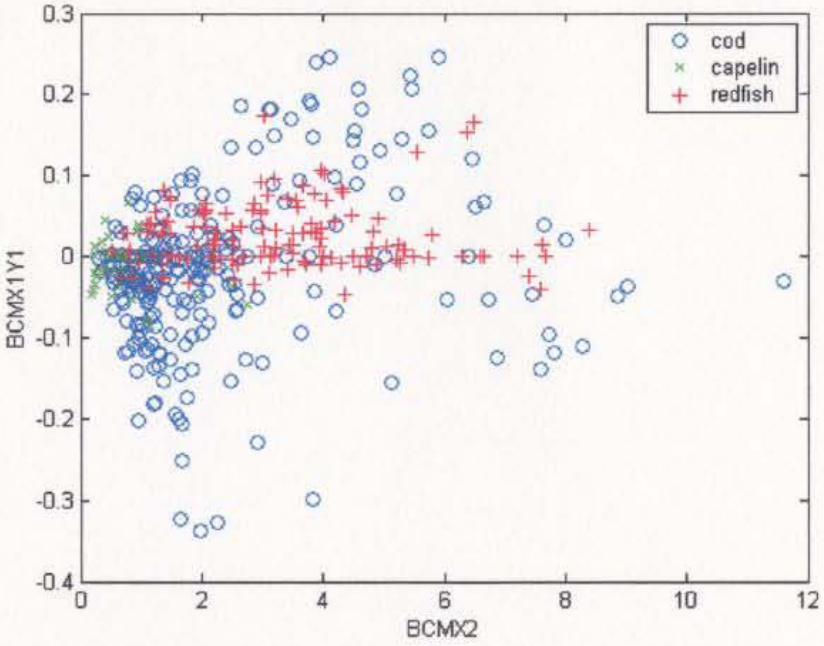


Figure 24: Feature plot for *BCM2* and *BCMX1Y1*

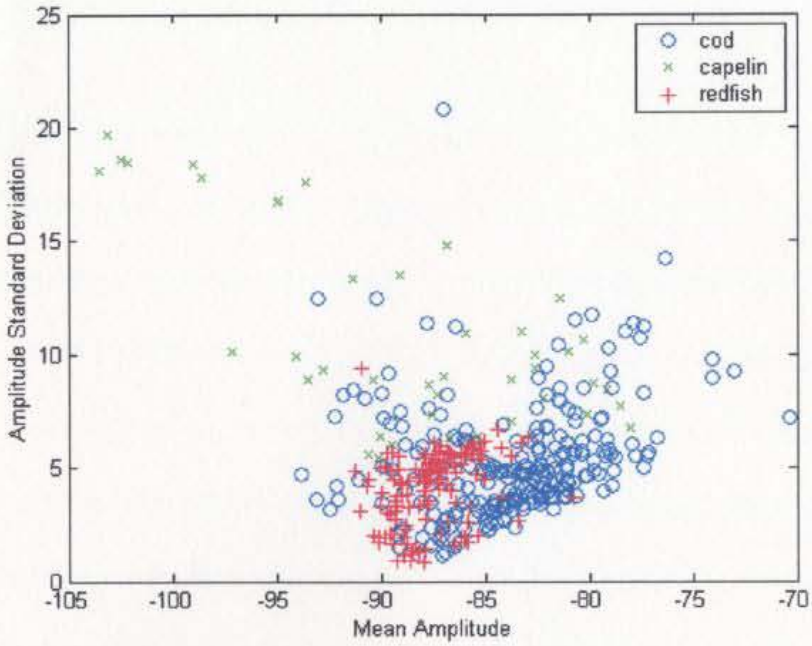


Figure 25: Feature plot for Amp_{mean} (dB) and Amp_{sd} (dB)

6.0 CONCLUSION

The potential benefits of automated fish identification for fisheries scientists and fishers are significant. Scientific acoustic surveys are destructive, expensive and labor intensive partially due to the requirement for ground truthing by way of hand-lining or trawling. For many years experienced scientists and fishers have been able to identify the shapes on their echograms but such judgements are subjective and unquantifiable. An automated system will allow for objective, non-destructive, fast, and repeatable species identification. For a commercial fishery a real-time fish identification system would be very beneficial from an environmental and economic point of view by helping to reduce by-catch. The technology could be adapted to allow commercial fishers to identify the species seen on their echo-sounders, allowing them to fish selectively by adjusting their fishing gear to avoid harvesting non-target species.

Based on the work done here and the work of others it is evident that echosounder technology, computing technology, and pattern recognition techniques have evolved to a point where automated species identification is technically feasible. For the three species studied here Atlantic capelin (*Mallotus villosus*), cod (*Gadus morhua*) and redfish (*Sebastes spp.*) the classification results produced by this work were very good (the best classifier was on average 96.8% correct) using a reasonable number of features. This classifier (configuration #2) was a dual node classifier where both nodes were 3-NN classifiers. The feature set used by the first node (for the identification of capelin) was made up by the

following six features: *Compactness, Minimum Amplitude, Distance from Object to Seabed, Central Moment X1Y1, Binarized Central Moment X2Y0*, and *Binarized Central Moment X1Y1*. The feature set used by the second node (for the identification of cod and redfish) was made up by the following five features: *Roughness, Mean Amplitude, Amplitude Standard Deviation, Distance from Object to Seabed*, and *Central Moment X0Y2*. This classifier provided the best overall classification and used on average the lowest number of features.

7.0 RECOMMENDATIONS

If this work it to be developed further, a number of areas need additional attention. Classification may be improved by using the Mahalanobis distance classifiers if the species data is further subdivided into subclasses of individual fish and schooling fish. The parametric nature of Mahalanobis distance classifiers means that they try to make generalizations about a population. This may have been made more difficult by grouping schools and individuals of one species into a single class. As well, a more sophisticated feature reduction technique should be implemented. The backward and forward sequential feature selection schemes used here cannot anticipate the interactions between features and likely did not find the optimum combination of features for each classifier tested. The "windowing" of fish, which was done manually, should also be automated. Given the recent increase in computer speeds, the allowable image size should be increased from the present limit of 512 x 512 pixels.

Having said in Section 6.0 Conclusion that the system could be further developed for general usage, it must realized that, as with any automated pattern recognition system applied to natural data, a great deal of ground truthed data is needed before a system can be ready for more general usage. Unfortunately for species identification of fish, data collection is very time consuming and expensive. To make a system for general scientific or commercial usage, data would have to be collected over at least a full year cycle, in a variety of areas, and for all species of scientific, environmental, and commercial interest. It should also be noted

that the classifiers would likely have to be retrained for data from different echosounders. Many amplitude and shape features are undoubtedly related to the echosounder properties such as the carrier frequency used (f), the pulse length (τ), beamwidth and sampling frequency. Nonetheless, given enough data, from a variety of echosounders, it may be possible to find features that are universally good descriptors for many species. Classification results will also depend on the quality of the echosounder system used. It is unlikely that consistently good classification results could be obtained from a low quality system. It is essential to have a system with a high signal to noise ratio and good signal stability.

8.0 REFERENCES

Brattey, J. and Morgan, M. J. (1996). "Temporal trends in the age and length at maturity of Atlantic cod (*Gadus morhua*) from NAFO Subdivision 3Ps," DFO Atlantic Fisheries Research Document 96/92.

Camp, L. (1970). *Underwater Acoustics*, Wiley-Interscience, Toronto, Ontario, Canada, 308 p.

Clay, C. S. and Medwin, H. (1977). *Acoustical Oceanography: Principles and Applications*, John Wiley Publishing, New York, New York, USA, 544 p.

Deuser, L., Middleton, D., Plemons, T. and Vaughan, J. (1979). "On the classification of underwater acoustic signals: II. Experimentation involving fish," *Journal of the Acoustical Society of America,* 65, pp. 444-445.

Diner, N., Weill, A., Coail, J.Y. and Coudeville, J.M. (1989). "INES-MOVIES: a new acoustic data acquisition and processing system," *ICES CM B:45, Fish Capture Committee,* 11 p.

Duda, R.O., and Hart, P.E. (1973). *Pattern Recognition and Scene Analysis*, John Wiley & Sons Inc., New York, New York, USA.

Fisher, F.H. and Simmonds, V.P. (1977). "Sound absorption in seawater," *Journal of the Acoustical Society of America*, Vol. 62, pp. 558-564.

Foote, K. G. (1987). "Fish Target Strengths for use in Echo Integrator Surveys," *Journal of the Acoustical Society of America*, Vol. 82, pp. 981-987.

Foote, K. G. (1980). "Effects of fish behaviour on echo energy: The need for measurements of orientation distributions," *Journal Conseil International pour l'Exploration de la Mer*, Vol. 93, pp. 193-201.

Foote, K. G., Aslen, A., and Nakken, O. (1986). "Measurement of fish target strength with a split-beam echo-sounder," *Journal of the Acoutical Society of America*, Vol. 80, pp. 612-621.

Gauthier, S., and Rose, G.A. (2001) "Target strength of encaged Atlantic redfish (*Sebastes spp.*)," *ICES Journal of Marine Science*. Vol.58, pp. 562-568.

Giryn, A., M. Rojewski, and Somla, R. (1981). "About the Possibility of Sea Creature Species Identification on the Basis of Applying Pattern Recognition to Echo-Sounder

Signals", Meeting on Hydroacoustical Methods for the Estimation of Marine Fish Populations, 25-29 June 1979. The Charles Stark Laboratory Inc., Cambridge, Mass. Vol. 2, Part A, pp. 455-465.

Giryn, A., Rojewski, M., and Somla, R. (1979). "About the possibility of sea creature species identification on the basis of applying pattern recognition to echo-sounder signals," *Proceedings of Meeting on Hydroacoustical Methods for the Estimation of Marine Fish Populations*, ed. J.B. Suomala, Charles Stark Draper Laboratory, Cambridge, MA, pp. 455-466.

Glasbey, C.A. and Horgan, G.W. (1995). *Image Analysis for the Biological Sciences*, John Wiley & Sons Inc., Toronto, Ontario, Canada, 218 p.

Gonzalez, R. C. and Woods, R. E. (1993). *Digital Image Processing*, Addison-Wesley Publishing Company Inc., Reading, Massachusetts, USA, 716 p.

Gregoire, F. (1999). "Capelin in the Estuary and Gulf of St. Lawrence," DFO-Science Stock Status Report B4 03

Hall, M.A. (2000). ""Correlation-based feature selection for discrete and numeric class machine learning," *Proceedings of the Seventeenth International Conference on Machine Learning*, San Francisco, CA. Morgan Kaufmann Publishers.

Holliday, D.V. (1977). "Extracting bio-physical information from the acoustic signatures of marine organisms," eds. N.R. Andersen and B.J. Zahuranec, *Oceanic sound scattering prediction*. Plenum Press, New York, New York, USA, pp. 619-624.

Jangaard, P.M. (1974). "The Capelin (*Mallotus villosus*): Biology, Distribution, Exploitation, Utilization, and Composition," Bulletin 186, Bulletin of the Fisheries Research Board of Canada, Environment Canada, Fisheries and Marine Service.

Johannesson, K. A. and Mitson, R. B. (1983). *Fisheries Acoustics: A Practical Manual for Aquatic Biomass Estimation*, FAO Fish. Tech. Pap. 249 p.

Lee, K. T., Lu, H. J., and Liao, C. H. (1996). "A study of the characteristics of fish shoals in coastal waters of Northeast Taiwan using an echo-signal image processing system," *Journal of the Fisheries Society of Taiwan*, Vol. 23, No. 3, pp. 181-193.

Lee, K.T., Liao, C.H. , Wu, L.J. , and Shih, W.H. (1990). "Classification of fish species by processing the hydroacoustic signal and canonical discriminant analysis," *Journal of the Fisheries Society of Taiwan*, Vol. 17, no. 3, pp. 161-170.

LeFeuvre, P., Rose, G. A., Gosine, R., Hale, R., Pearson, W. and Khan, R. (2000). "Acoustic

species identification in the Northwest Atlantic using digital image processing," *Fisheries Research*, 47 (2000), Elsevier Publishing, pp. 137-147.

LeFeuvre, P., Khan, R., Pike, C., Gosine, R., and Rose, G. (1996). "Development of Analytical Techniques and Software to Enable Taxonomic Identification of Acoustic Signals - Progress Report, February 1996," C-CORE Contract Report for Canadian Centre for Fisheries Innovation.

Love, R.H., (1971). "Measurement of fish target strength: a review," Fish Bull. 69, pp. 703-715.

Lu, H. J. and Lee, K. T. (1995). "Species identification of fish shoals from echograms by an echo-signal processing system," *Fisheries Research*, Vol 24, No. 2, pp. 99-111.

MacLennan, D.N., and Holliday, D.V. (1996). "Fisheries and plankton acoustics: past, present, and future," *ICES Journal of Marine Science,* Vol. 53, pp. 513-516.

MacLennan, D.N. and E.J. Simmonds. (1992). *Fisheries Acoustics*, Chapman & Hall, London, UK, 336 p.

Magand, F. (1994). "Fish Species Discrimination Using a Wideband Echosounder", *Proceedings of the Second European Conference on Underwater Acoustics*, July 4-8, 1994. Lyngby, Denmark. Vol. 2, pp. 821-826.

Magand, F., and Zakharia, M.E. (1992). "Fish Echo Classification Using an A Priori Physical Model: Advantages and Limitations," *Proceedings of the European Conference on Underwater Acoustics*, Luxembourg, pp. 129-132.

Midttun, L. (1984). "Fish and Other Organisms as Acoustic Targets," *Rapports et Proces - Verbaux des Reunions Conseil International pour l'Exporation de la Mer*, Vol 184, pp. 25-33.

Midttun, L. and Hoff, I. (1962). "Measurement of the reflection of sound by fish," Fiskeridir. Skr. Sev. Havunders, Vol. 13, pp. 1-18.

Misund, O. A. (1997). "Underwater Acoustics in Marine Fisheries and Fisheries Research," *Reviews in Fish Biology and Fisheries*, Vol. 7, no.1, pp. 22-34.

Miyashita, K., Aoki, I. , Seno, K., Taki, K., and Ogishima, T. (1997). "Acoustic Identification of Isada Krill, *Euphausia pacifica*, off the Sanriku Coast, North-eastern Japan," *Fisheries Oceanography*, Vol. 6, n. 4, pp.266-271.

Nakken, O. (1998). "Acoustic methods in studies of fish ecology," Opening Lecture, ICES

Annual Science Conference, Libson, September 16, 1998

Nakken, O. and Olsen, K. (1977). "Target-strength measurements of fish," *Rapports et Proces - Verbaux des Reunions Conseil International pour l'Exporation de la Mer*, Vol. 170, pp. 52-69.

Pilanowski, R.A., Morse, W.W., Berrien, P.L., Johnson, D.L., and D.G. McMillan, (1999). "Redfish, *Sebastes spp.*, Life History and Habitat Characteristics." National Marine Fisheries Service, NOAA Technical Memorandum NMFS-NE-132

Power, D. (1998). "October 1998 Status Of Redfish Stocks in the Northwest Atlantic Redfish in Units 1 2 and 3 and in Division 3O," DFO Science Stock Status Report A1 01.

Reid, D.G. and E.J. Simmonds, (1993). "Image analysis techniques for the study of fish school structure from acoustic survey data," *Canadian Journal of Fisheries and Aquatic Sciences,* Vol. 50, pp. 886-893.

Richards, L.J., Kieser, R., Mulligan, T.J., and Candy J.R. (1991). "Classification of Fish Assemblages Based on Echo Integration Surveys," *Canadian Journal of Fisheries Aquatic Science*, Vol 48, pp. 1264-1272.

Rose, G.A. (1998). "Acoustic target strength of capelin in Newfoundland waters," *ICES Journal of Marine Science*, Vol. 55, pp. 918-923.

Rose, G.A. and Porter, D.R. (1996). "Target strength studies on Atlantic cod (*Gadus morhua*) in Newfoundland waters," *ICES Journal of Marine Science*, Vol. 53, pp. 259-265.

Rose, G.A., (1992). "A review of problems and new directions in the application of fisheries acoustics on the Canadian East Coast," *Fisheries Research*, Vol. 14, pp. 105-128.

Rose, G.A. and Leggett, W.C. (1988). "Hydroacoustic signal classification of fish schools by species," *Canadian Journal of Fisheries and Aquatic Sciences,* Vol. 45, pp. 597-604.

Russ, J.C. (1995). *The Image Processing Handbook*, CRC Press Inc., Boca Raton, Florida, USA.

Scalabrin, C., Diner, N., Weill, A., Hillion, A., and Mouchot, C. C. (1996). "Narrowband acoustic identification of monospecific fish shoals," *ICES Journal of Marine Science*, Vol. 53, pp. 181-188.

Scalabrin, C., Diner, N., and Wiell, A. (1994). "Automatic Shoal Recognition and

Classification Based on MOVIES-B Software," *Proceedings of Oceans'94*, Brest, France. Vol. 2, pp.319-324.

Simmonds, E. J., and Armstrong, F. (1990). "A wideband echo-sounder: measurements on cod, saithe, herring, and mackerel from 27 to 54 kHz," *Rapports et Proces-Verbaux des Reunions du Conseil International pour l'Exploration de la Mer,* Vol. 189, pp. 381-387.

Simmonds, E. J., Armstrong, F., and Copland, PH.J. (1996). "Species identification using wideband backscatter with neural network and discriminant analysis," *ICES Journal of Marine Science*, Vol. 53, No. 2, pp.189-195.

Souid, P. (1988). "Automatisation de la description et de la classification des detections acoustiques de bancs de poissons pelagiques pour leur identification". *PhD thesis*, Universite d'Aix Marseille II, France, 225 p.

Urick, R. J. (1983). *Principles of Underwater Sound*, McGraw-Hill Book Company, Montreal, Quebec, Canada, 423 p.

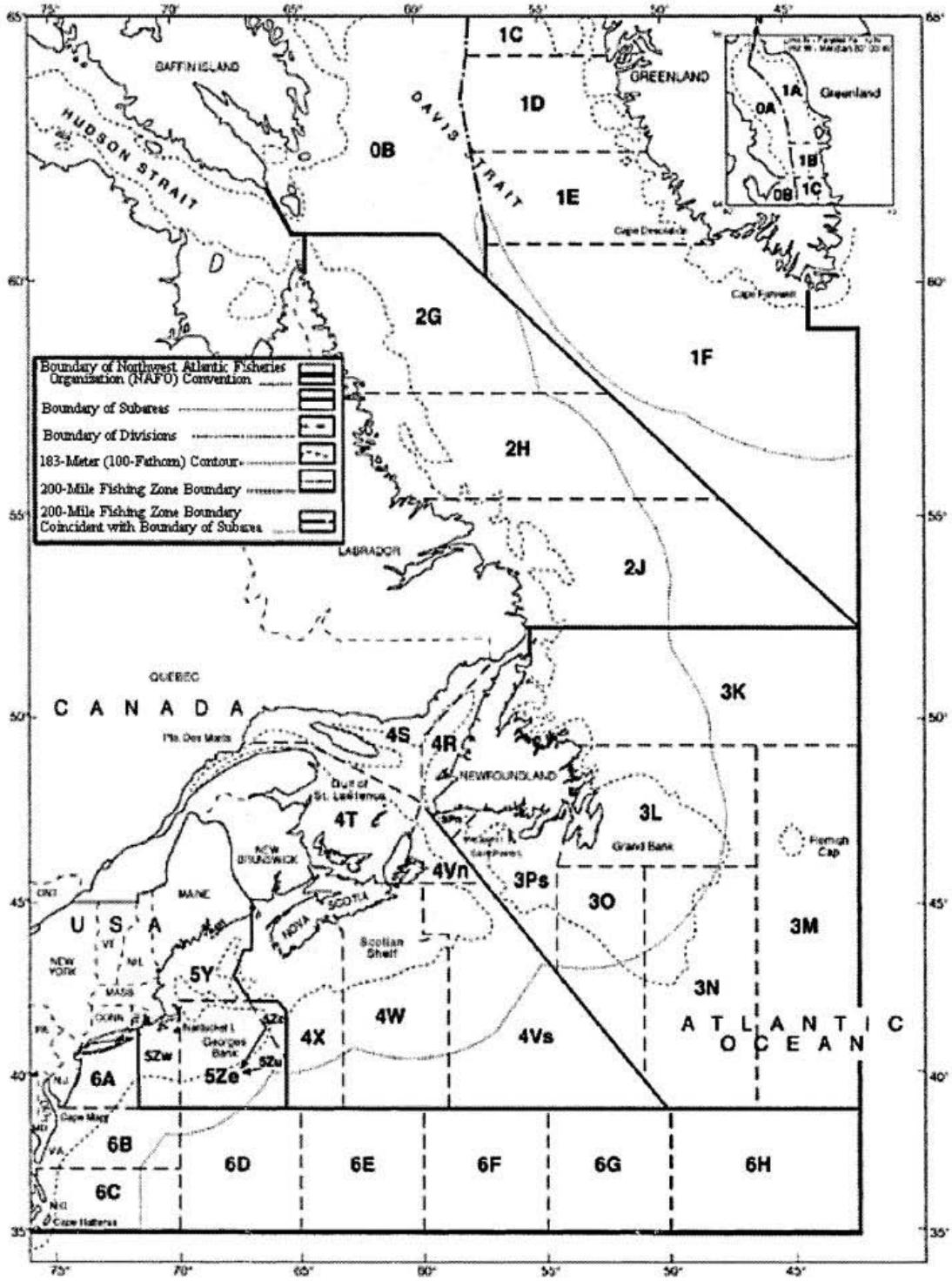
Weill, A., Scalabrin, C., and Diner, N. (1993). "MOVIES-B: an acoustic detection description software. Application to shoal species classification," *Aquatic Living Resources*, Vol. 6, pp. 255-267.

Weiss, S. M., and Kulikowski, C. A. (1991). *Computer Systems that Learn*. Morgan Kaufmann Publishers Inc., Palo Alto, California, USA, 223 p.

Zakharia, M. E., Magand, F., Hetroit, F., and Diner, N. (1996). "Wideband sounder for fish species identification at sea," *ICES Journal of Marine Science*, Vol. 53, pp. 203-208.

Zakharia, M. E. and Sessarego, J. P. (1982). "Sonar target classification using coherent echo processing," *Proceedings from the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Paris, France. pp. 331-334.

Appendix A
Data Collection Area Maps



Map of NAFO Convention Area (www.nafo.ca/imap/map.htm)

Trinity Bay

Placentia Bay

Appendix B
Acoustic Calculations

B. 1. Sound Velocity Calculation

The following calculation is used to estimate the speed of sound in sea water based on knowledge of environmental conditions (Camp, 1970):

$$c = 1449 + 4.6T - 0.055T^2 + 0.0003T^3 + (1.39 - 0.012T)(S - 35) + 0.017d$$

where: c = velocity in metres/second
T = water temperature in degrees in Celsius
S = salinity in parts per thousand
d = depth in metres

B. 2. Absorption Coefficient Calculation (Fisher and Simmond, 1997):

Sound absorption (α) in seawater can be approximated using the following. Since the change in absorption due to water pressure at depths less than 1000 m is negligible, pressure has been left out of the following equation, which assumes pH = 8.

$$\alpha = \frac{A_1 f_1 f^2}{f_1^2 + f^2} + \frac{A_2 f_2 f^2}{f_2^2 + f^2} + A_3 f^2$$

where:

$$A_1 = 1.03x10^{-8} + 2.36x10^{-10}T - 5.22x10^{-12}T^2$$

$$A_2 = \frac{S}{35}(5.62x10^{-8} + 7.52x10^{-10}T)$$

$$A_3 = (55.9 - 2.37T + 4.77x10^{-2}T^2 - 3.48x10^{-4}T^3)x10^{-15}$$

$$f_1 = 1.32x10^3(T + 273.1)\exp^{\frac{-1700}{T+273.1}}$$

$$f_2 = 1.55x10^7(T + 273.1)\exp^{\frac{-3052}{T+273.1}}$$

T = water temperature in degrees in Celsius
f = sound frequency in Hz

S = salinity in parts per thousand

B. 3. Time Varied Gain

In order measure an acoustic target, it is necessary to correct the received echo for the effects of spherical spreading and absorption losses. The correction for these losses is done by applying time varied gain (*TVG*) to the receiving amplifier which until recently was done using analog circuitry. For single fish targets a *TVG* of 40 $\log(r)$ + $2\alpha r$ is used to compensate for two-way spreading and absorption losses (to and from the target) where α is absorption coefficient [see B.2. for the estimation of α]. For densely schooling fish a *TVG* of 20 $\log(r)$ + $2\alpha r$ is used. (MacLennan and Simmonds, 1992)

B. 4. Target Strength

Target Strength (TS) is used to describe the acoustic reflectivity of targets. The reflectivity is defined as the ratio of the reflected intensity at 1 m from the target (I_2) and the incident intensity (I_1). I_2 is proportional to I_1, therefore TS, given in dB, is not quoted with reference to a specific pressure level. (MacLennan and Simmonds, 1992)

$$TS = 10 \log (I_2/I_1)$$

Appendix C
Echo-sounder Parameters

Echo-sounder Parameters

BioSonics DT4000

Source Level	222 dB
Receiver Sensitivity	-52 dB
Lower Data Threshold (used during data collection)	-130 dB
Transducer Frequency	38 KHz
Ping Rate	5 Hz
Pulse Width	0.4 ms
Major Axis Beamwidth	6°
Minor Axis Beamwidth	6°

Appendix D
Matlab© Code

```
%   classC1.m
%
%     Patricia LeFeuvre
%
%   Thesis classifier configuration #1
%   1 node:
%   3NN - separate cod, capelin and redfish

clear,

% Feature Key
%1    Area
%2    Perim
%3    Compac
%4    Rough
%5    Width
%6    Height
%7    Axis
%8    Elong
%9    Mean
%10   Max
%11   Min
%12   Sigma
%13   depth to centroid
%14   depth to top
%15   Dist off bottom
%16   holes
%17   NA
%18   NA
%19   NA
%20   mcx0y2
%21   mcx2y0
%22   mcx1y1
%23   mcx0y2bin
%24   mcx2y0bin
%25   mcx1y1bin

clear,
load capelin.txt;
load cod.txt;
load redfish.txt;

%normalize all of the data
MN_red = min(redfish);
MN_capelin = min(capelin);
MN_cod = min(cod);
MN = min(min(MN_red,MN_capelin),MN_cod);

MX_red = max(redfish);
MX_capelin = max(capelin);
MX_cod = max(cod);
MX = max(max(MX_red,MX_capelin),MX_cod)-MN;

for i = 1:length(redfish)
      norm_red(i,1:25) = (redfish(i,1:25) - MN)./MX;
end

for i = 1:length(capelin)
```

```
      norm_capelin(i,1:25) = (capelin(i,1:25) - MN)./MX;
end

for i = 1:length(cod)
      norm_cod(i,1:25) = (cod(i,1:25) - MN)./MX;
end

alldata = [norm_cod; norm_capelin; norm_red];

num_cod = length(norm_cod);
num_capelin = length(norm_capelin);
num_redfish = length(norm_red);


%% CLASSIFIER Configuration #1
% After depth feature removal, Factor Analysis and SBS
keep1 =    [9 10 11 12 15 20 22 24 25];
keepList = 1:25 * 0;
keepList(keep1) = 1;

cod_called_red = 0;
cod_called_cod = 0;
cod_called_cap = 0;
capelin_called_red = 0;
capelin_called_cod = 0;
capelin_called_cap = 0;
redfish_called_red = 0;
redfish_called_cod = 0;
redfish_called_cap = 0;

for i = 1:length(alldata)
      data = [alldata(i,keep1)];

      for j = 1:length(alldata)
            if i ~= j
                  tmp = data - alldata(j,keep1);
                  dist(j) = sqrt(tmp*tmp');
            else
                  dist(j) = 25;
            end
      end

      [sortedDist, Index] = sort(dist);

      if ( length(find(Index(1:3) <= num_cod )) >= 2) % 2/3 were cod
            called = 1; % 1 = cod;
      elseif ( length(find( (Index(1:3) > num_cod) & (Index(1:3) <=
(num_cod+num_capelin))) ) >= 2) % 2/3 were capelin
            called = 2; % 2 = capelin
      elseif ( length(find(Index(1:3) > (num_cod+num_capelin))) >= 2) % 2/3 were
redfish
            called = 3; % 3 = redfish
      else
            %3 way tie - use the distance to the nearest neighbour
            if (Index(1) <= num_cod)
                  called = 1;
            elseif (Index(1) <= (num_cod+num_capelin))
                  called = 2;
```

```
            else
                  called = 3;
            end
      end


      if called == 1
            if (i <= num_cod)
                  cod_called_cod = cod_called_cod + 1;
            elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                  capelin_called_cod = capelin_called_cod + 1;
            else
                  redfish_called_cod = redfish_called_cod + 1;
            end

      elseif called == 2
            if (i <= num_cod)
                  cod_called_cap = cod_called_cap + 1;
            elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                  capelin_called_cap = capelin_called_cap + 1;
            else
                  redfish_called_cap = redfish_called_cap + 1;
            end

      else
            if (i <= num_cod)
                  cod_called_red = cod_called_red + 1;
            elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                  capelin_called_red = capelin_called_red + 1;
            else
                  redfish_called_red = redfish_called_red + 1;
            end

      end
end

fprintf('Number of Features = %d\nFeature List = ',length(keep1));
fprintf('%d  ', keep1);
fprintf('\n\t\t\t\tPREDICTED CLASS\n');
fprintf('\t\tCOD\t\tCAPELIN\t\tREDFISH\n');

fprintf('COD\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n',...
cod_called_cod,cod_called_cod/num_cod*100,...
cod_called_cap,cod_called_cap/num_cod*100,...
cod_called_red,cod_called_red/num_cod*100);

fprintf('CAPELIN\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n',...
capelin_called_cod,capelin_called_cod/num_capelin*100,...
capelin_called_cap,capelin_called_cap/num_capelin*100,...
capelin_called_red,capelin_called_red/num_capelin*100);

fprintf('REDFISH\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n\n',...
redfish_called_cod,redfish_called_cod/num_redfish*100,...
redfish_called_cap,redfish_called_cap/num_redfish*100,...
redfish_called_red,redfish_called_red/num_redfish*100);
```

```
%   classC2.m
%
%   Patricia LeFeuvre
%   Feb 25, 2001
%
%   Thesis classifier configuration #2
%   2 nodes:
%   node #1 - 3NN - extract cod
%   node #2 - 3NN - separate capelin and redfish

clear,
load capelin.txt;
load cod.txt;
load redfish.txt;

%normalize all of the data
MN_red = min(redfish);
MN_capelin = min(capelin);
MN_cod = min(cod);
MN = min(min(MN_red,MN_capelin),MN_cod);

MX_red = max(redfish);
MX_capelin = max(capelin);
MX_cod = max(cod);
MX = max(max(MX_red,MX_capelin),MX_cod)-MN;

for i = 1:length(redfish)
      norm_red(i,1:25) = (redfish(i,1:25) - MN)./MX;
end

for i = 1:length(capelin)
      norm_capelin(i,1:25) = (capelin(i,1:25) - MN)./MX;
end

for i = 1:length(cod)
      norm_cod(i,1:25) = (cod(i,1:25) - MN)./MX;
end

% Classifier #2
alldata = [norm_cod; norm_capelin; norm_red];
node2data = [norm_capelin; norm_red];

cod_called_red = 0;
cod_called_cod = 0;
cod_called_cap = 0;
capelin_called_red = 0;
capelin_called_cod = 0;
capelin_called_cap = 0;
redfish_called_red = 0;
redfish_called_cod = 0;
redfish_called_cap = 0;

num_cod = length(norm_cod);
num_capelin = length(norm_capelin);
num_redfish = length(norm_red);


% list of features selected
```

```
% Node 1 - extract cod
keep1 =    [3 4 6 9 11 12 15 22 24 25];

% Node 2 - capelin vs. redfish
keep2 =    [3 9 11 15 22 24 25];

for i = 1:length(alldata)

      data = [alldata(i,keep1)];
      for j = 1:length(alldata)
            if i ~= j
                  tmp = data - alldata(j,keep1);
                  dist(j) = sqrt(tmp*tmp');
            else
                  dist(j) = 25;
            end
      end

      [sortedDist, Index] = sort(dist);

      if ( length(find(Index(1:3) <= num_cod )) >= 2)
            called = 1; % 1 = cod;
   else
      % Node 2
      data = [alldata(i,keep2)];
        sameCount = 0;
      for j = 1:length(node2data)
                        tmp = data - node2data(j,keep2);
            dist(j) = sqrt(tmp*tmp');
            if (dist(j) == 0) % the same point so don't use
               dist(j) == 25;
               sameCount = sameCount + 1;
               if sameCount > 1
                  fprintf('Error - too many identical points ');
               end

            end
      end
      [sortedDist, Index] = sort(dist);

            if ( length(find(Index(1:3) <= num_capelin )) >= 2) % 2/3 were capelin
                  called = 2; % 2 = capelin
          else
                called = 3; % 3 = redfish
        end

      end  % end of Node 2


      if called == 1
            if (i <= num_cod)
                  cod_called_cod = cod_called_cod + 1;
            elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                  capelin_called_cod = capelin_called_cod + 1;
            else
                  redfish_called_cod = redfish_called_cod + 1;
      end
```

```
      elseif called == 2
            if (i <= num_cod)
                  cod_called_cap = cod_called_cap + 1;
            elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                  capelin_called_cap = capelin_called_cap + 1;
            else
                  redfish_called_cap = redfish_called_cap + 1;
            end

      else
            if (i <= num_cod)
                  cod_called_red = cod_called_red + 1;
            elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                  capelin_called_red = capelin_called_red + 1;
            else
                  redfish_called_red = redfish_called_red + 1;
            end
   end
end


fprintf('Node #1 Number of Features = %d\nFeature List = ',length(keep1));
fprintf('%d  ', keep1);

fprintf('\nNode #2 Number of Features = %d\nFeature List = ',length(keep2));
fprintf('%d  ', keep2);

fprintf('\n\t\t\t\tPREDICTED CLASS\n');
fprintf('\t\tCOD\t\tCAPELIN\t\tREDFISH\n');

fprintf('COD\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n',...
cod_called_cod,cod_called_cod/num_cod*100,...
cod_called_cap,cod_called_cap/num_cod*100,...
cod_called_red,cod_called_red/num_cod*100);

fprintf('CAPELIN\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n',...
capelin_called_cod,capelin_called_cod/num_capelin*100,...
capelin_called_cap,capelin_called_cap/num_capelin*100,...
capelin_called_red,capelin_called_red/num_capelin*100);

fprintf('REDFISH\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n\n',...
redfish_called_cod,redfish_called_cod/num_redfish*100,...
redfish_called_cap,redfish_called_cap/num_redfish*100,...
redfish_called_red,redfish_called_red/num_redfish*100);
```

```
%   classC3.m
%
%     Patricia LeFeuvre
%     Feb 25, 2001
%   Thesis classifier configuration #3
%   2 nodes:
%   node #1 - 3NN - extract capelin
%   node #2 - 3NN - separate cod and redfish

clear,
load capelin.txt;
load cod.txt;
load redfish.txt;

%normalize all of the data
MN_red = min(redfish);
MN_capelin = min(capelin);
MN_cod = min(cod);
MN = min(min(MN_red,MN_capelin),MN_cod);

MX_red = max(redfish);
MX_capelin = max(capelin);
MX_cod = max(cod);
MX = max(max(MX_red,MX_capelin),MX_cod)-MN;

for i = 1:length(redfish)
      norm_red(i,1:25) = (redfish(i,1:25) - MN)./MX;
end

for i = 1:length(capelin)
      norm_capelin(i,1:25) = (capelin(i,1:25) - MN)./MX;
end

for i = 1:length(cod)
      norm_cod(i,1:25) = (cod(i,1:25) - MN)./MX;
end

% Classifier #2
alldata = [norm_cod; norm_capelin; norm_red];
node2data = [norm_cod; norm_red];

cod_called_red = 0;
cod_called_cod = 0;
cod_called_cap = 0;
capelin_called_red = 0;
capelin_called_cod = 0;
capelin_called_cap = 0;
redfish_called_red = 0;
redfish_called_cod = 0;
redfish_called_cap = 0;

num_cod = length(norm_cod);
num_capelin = length(norm_capelin);
num_redfish = length(norm_red);


% list of features selected
```

```
% Node 1 - extract capelin
keep1 =    [3 11 15 22 25];

% Node 2 - cod vs. redfish
keep2 =    [4 9 12 15 20];

for i = 1:length(alldata)
      data = [alldata(i,keep1)];
      for j = 1:length(alldata)
            if i ~= j
                  tmp = data - alldata(j,keep1);
                  dist(j) = sqrt(tmp*tmp');
            else
                  dist(j) = 25;
            end
      end

      [sortedDist, Index] = sort(dist);

      if ( length(find( (Index(1:3) > num_cod) & (Index(1:3) <=
(num_cod+num_capelin))) ) >= 2) % 2/3 were capelin
                  called = 2; % 2 = capelin;
    else
      % Node 2
        data = [alldata(i,keep2)];
          sameCount = 0;
        for j = 1:length(node2data)
                  tmp = data - node2data(j,keep2);
            dist2(j) = sqrt(tmp*tmp');
            if (dist2(j) == 0) % the same point so don't use
               dist2(j) == 25;
               sameCount = sameCount + 1;
               if sameCount > 1
                  fprintf('Error - too many identical points ');
               end
            end
        end
        [sortedDist, Index] = sort(dist2);

            if ( length(find(Index(1:3) <= num_cod )) >= 2) % 2/3 were cod
                  called = 1; % 1 = cod
          else
                called = 3; % 3 = redfish
        end
    end  % end of Node 2

      if called == 1
            if (i <= num_cod)
                  cod_called_cod = cod_called_cod + 1;
            elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                  capelin_called_cod = capelin_called_cod + 1;
            else
                  redfish_called_cod = redfish_called_cod + 1;
      end

      elseif called == 2
            if (i <= num_cod)
                  cod_called_cap = cod_called_cap + 1;
```

```
            elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                  capelin_called_cap = capelin_called_cap + 1;
            else
                  redfish_called_cap = redfish_called_cap + 1;
            end

      else
            if (i <= num_cod)
                  cod_called_red = cod_called_red + 1;
            elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                  capelin_called_red = capelin_called_red + 1;
            else
                  redfish_called_red = redfish_called_red + 1;
            end
   end

end


fprintf('Node #1 Number of Features = %d\nFeature List = ',length(keep1));
fprintf('%d  ', keep1);

fprintf('\nNode #2 Number of Features = %d\nFeature List = ',length(keep2));
fprintf('%d  ', keep2);

fprintf('\n\t\t\t\tPREDICTED CLASS\n');
fprintf('\t\tCOD\t\tCAPELIN\t\tREDFISH\n');

fprintf('COD\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n',...
cod_called_cod,cod_called_cod/num_cod*100,...
cod_called_cap,cod_called_cap/num_cod*100,...
cod_called_red,cod_called_red/num_cod*100);

fprintf('CAPELIN\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n',...
capelin_called_cod,capelin_called_cod/num_capelin*100,...
capelin_called_cap,capelin_called_cap/num_capelin*100,...
capelin_called_red,capelin_called_red/num_capelin*100);

fprintf('REDFISH\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n\n',...
redfish_called_cod,redfish_called_cod/num_redfish*100,...
redfish_called_cap,redfish_called_cap/num_redfish*100,...
redfish_called_red,redfish_called_red/num_redfish*100);
```

```
% classC4.m
%
%     Patricia LeFeuvre
%     Feb 26, 2001
%   Thesis classifier configuration #4
%   2 nodes:
%   node #1 - 3NN - extract redfish
%   node #2 - 3NN - separate cod and capelin
clear,

load capelin.txt;
load cod.txt;
load redfish.txt;

%normalize all of the data
MN_red = min(redfish);
MN_capelin = min(capelin);
MN_cod = min(cod);
MN = min(min(MN_red,MN_capelin),MN_cod);

MX_red = max(redfish);
MX_capelin = max(capelin);
MX_cod = max(cod);
MX = max(max(MX_red,MX_capelin),MX_cod)-MN;

for i = 1:length(redfish)
      norm_red(i,1:25) = (redfish(i,1:25) - MN)./MX;
end

for i = 1:length(capelin)
      norm_capelin(i,1:25) = (capelin(i,1:25) - MN)./MX;
end

for i = 1:length(cod)
      norm_cod(i,1:25) = (cod(i,1:25) - MN)./MX;
end

alldata = [norm_cod; norm_capelin; norm_red];
node2data = [norm_cod; norm_capelin];

cod_called_red = 0;
cod_called_cod = 0;
cod_called_cap = 0;
capelin_called_red = 0;
capelin_called_cod = 0;
capelin_called_cap = 0;
redfish_called_red = 0;
redfish_called_cod = 0;
redfish_called_cap = 0;

num_cod = length(norm_cod);
num_capelin = length(norm_capelin);
num_redfish = length(norm_red);


% selected features
% Node 1 - extract redfish
keep1 =    [4 9 10 11 12 15 16 20 24 25];
```

```
%Node 2 - cod vs. capelin
keep2 =    [3 4 6 7 9 12 15 20];


for i = 1:length(alldata)

      data = [alldata(i,keep1)];
      for j = 1:length(alldata)
            if i ~= j
                  tmp = data - alldata(j,keep1);
                  dist(j) = sqrt(tmp*tmp');
            else
                  dist(j) = 25;
            end
      end

      [sortedDist, Index] = sort(dist);

    if ( length(find(Index(1:3) > (num_cod+num_capelin))) >= 2)% 2/3 were redfish
          called = 3; % 3 = redfish;
    else
        % Node 2
        data = [alldata(i,keep2)];
            sameCount = 0;
        for j = 1:length(node2data)
                  tmp = data - node2data(j,keep2);
            dist2(j) = sqrt(tmp*tmp');
            if (dist2(j) == 0) % the same point so don't use
               dist2(j) == 25;
               sameCount = sameCount + 1;
               if sameCount > 1
                  fprintf('Error - too many identical points\n ');
                  pause;
               end
           end
        end
        [sortedDist, Index] = sort(dist2);

        if ( length(find(Index(1:3) <= num_cod )) >= 2) % 2/3 were cod
                called = 1; % 1 = cod
            else
                called = 2; % 2 = capelin
        end
    end  % end of Node 2

      if called == 1
            if (i <= num_cod)
                  cod_called_cod = cod_called_cod + 1;
            elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                  capelin_called_cod = capelin_called_cod + 1;
            else
                  redfish_called_cod = redfish_called_cod + 1;
      end

      elseif called == 2
            if (i <= num_cod)
                  cod_called_cap = cod_called_cap + 1;
            elseif (i > num_cod) & (i <= (num_cod + num_capelin))
```

```
                capelin_called_cap = capelin_called_cap + 1;
            else
                redfish_called_cap = redfish_called_cap + 1;
            end

        else
            if (i <= num_cod)
                cod_called_red = cod_called_red + 1;
            elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                capelin_called_red = capelin_called_red + 1;
            else
                redfish_called_red = redfish_called_red + 1;
            end
    end
end

fprintf('Node #1 Number of Features = %d\nFeature List = ',length(keep1));
fprintf('%d  ', keep1);
fprintf('\nNode #2 Number of Features = %d\nFeature List = ',length(keep2));
fprintf('%d  ', keep2);

fprintf('\n\t\t\t\tPREDICTED CLASS\n');
fprintf('\t\tCOD\t\tCAPELIN\t\tREDFISH\n');

fprintf('COD\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n',...
cod_called_cod,cod_called_cod/num_cod*100,...
cod_called_cap,cod_called_cap/num_cod*100,...
cod_called_red,cod_called_red/num_cod*100);

fprintf('CAPELIN\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n',...
capelin_called_cod,capelin_called_cod/num_capelin*100,...
capelin_called_cap,capelin_called_cap/num_capelin*100,...
capelin_called_red,capelin_called_red/num_capelin*100);

fprintf('REDFISH\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n\n',...
redfish_called_cod,redfish_called_cod/num_redfish*100,...
redfish_called_cap,redfish_called_cap/num_redfish*100,...
redfish_called_red,redfish_called_red/num_redfish*100);
```

```
%   classC5.m
%
%     Patricia LeFeuvre
%     Feb 25, 2001
%   Thesis classifier configuration #5
%   1 node:
%   MAH - separate cod, capelin and redfish

clear,
load capelin.txt;
load cod.txt;
load redfish.txt;

% Normalize
MN_red = min(redfish);
MN_capelin = min(capelin);
MN_cod = min(cod);
MN = min(min(MN_red,MN_capelin),MN_cod);

MX_red = max(redfish);
MX_capelin = max(capelin);
MX_cod = max(cod);
MX = max(max(MX_red,MX_capelin),MX_cod)-MN;

for i = 1:length(redfish)
      norm_red(i,1:25) = (redfish(i,1:25) - MN)./MX;
end

for i = 1:length(capelin)
      norm_capelin(i,1:25) = (capelin(i,1:25) - MN)./MX;
end

for i = 1:length(cod)
      norm_cod(i,1:25) = (cod(i,1:25) - MN)./MX;
end

alldata = [norm_cod; norm_capelin; norm_red];


num_cod = length(norm_cod);
num_capelin = length(norm_capelin);
num_redfish = length(norm_red);

% list of features still in the running
keep1 =    [2 3 4 6 7 11 12 15];

% look at performance vs num features used - look for over training
%keep1 =    [1 2 3 4 5 6 7 9 10 11 12 15 16 20 22 24 25];   % perf = 86.4285,
num_feats = 17
%keep1 =    [1 2 3 4 5 6 7 9 10 11 12 15 16 20    24 25];   % perf = 88.1698,
num_feats = 16
%keep1 =    [1 2 3 4 5 6 7   10 11 12 15 16 20    24 25];   % perf = 89.4136,
num_feats = 15
%keep1 =    [1 2 3 4 5 6 7      11 12 15 16 20    24 25];   % perf = 89.2187,
num_feats = 14
%keep1 =    [1 2 3 4 5 6 7      11 12 15 16 20       25];   % perf = 90.1136,
num_feats = 13
%keep1 =    [1 2 3 4 5 6 7      11 12 15    20       25];   % perf = 88.9225,
```

```
num_feats = 12
%keep1 =    [1 2 3 4 5 6 7      11 12 15    20          ];   % perf = 91.1075,
num_feats = 11
%keep1 =    [1 2 3 4 5 6 7      11 12 15                ];   % perf = 91.0700,
num_feats = 10
%keep1 =    [  2 3 4 5 6 7      11 12 15                ];   % perf = 91.7150,
num_feats = 9
%keep1 =    [  2 3 4   6 7      11 12 15                ];   % perf = 91.7150,
num_feats = 8
%keep1 =    [    3 4   6 7      11 12 15                ];   % perf = 91.7239,
num_feats = 7
%keep1 =    [    3 4     7      11 12 15                ];   % perf = 90.2852,
num_feats = 6
%keep1 =    [      4     7      11 12 15                ];   % perf = 89.4465,
num_feats = 5
%keep1 =    [      4     7      11    15                ];   % perf = 87.7216,
num_feats = 4
%keep1 =    [      4     7            15                ];   % perf = 84.1180,
num_feats = 3
%keep1 =    [      4                  15                ];   % perf = 81.6040,
num_feats = 2
%keep1 =    [                         15                ];   % perf = 67.0804,
num_feats = 1


keepList = 1:25 * 0;
keepList(keep1) = 1;

%% CLASSIFIER #5 - Distinguish cod from capelin from Redfish
training_cod = norm_cod;                    %[norm_cod(:,keep1)];
training_capelin = norm_capelin; %[norm_capelin(:,keep1)];
training_red = norm_red;                    %[norm_red(:,keep1)];



cod_called_red = 0;
cod_called_cod = 0;
cod_called_cap = 0;
capelin_called_red = 0;
capelin_called_cod = 0;
capelin_called_cap = 0;
redfish_called_red = 0;
redfish_called_cod = 0;
redfish_called_cap = 0;


for i = 1:length(alldata)

   M_cod = mean(training_cod(:,keep1));
   M_capelin = mean(training_capelin(:,keep1));
   M_red = mean(training_red(:,keep1));

   K_cod = cov(training_cod(:,keep1));
   K_capelin = cov(training_capelin(:,keep1));
   K_red = cov(training_red(:,keep1));

   % leave out the sample we are testing
   if( i<=num_cod)
```

```
      M_cod = mean(training_cod(find(1:num_cod~=i),keep1));
      K_cod = cov(training_cod(find(1:num_cod~=i),keep1));
   elseif (i > num_cod) & (i <= (num_cod + num_capelin))
      M_capelin = mean(training_capelin(find(1:num_capelin~=(i-num_cod)),keep1));
      K_capelin = cov(training_capelin(find(1:num_capelin~=(i-num_cod)),keep1));
   else
      M_red =
mean(training_red(find(1:num_redfish~=(i-num_cod-num_capelin)),keep1));
      K_red =
cov(training_red(find(1:num_redfish~=(i-num_cod-num_capelin)),keep1));
   end


   data = [alldata(i,keep1)];
   dist_cod = (data - M_cod) * inv(K_cod) * (data - M_cod)';
   dist_capelin = (data - M_capelin) * inv(K_capelin) * (data - M_capelin)';
   dist_red = (data - M_red) * inv(K_red) * (data - M_red)';
   if ( (dist_cod < dist_capelin) & (dist_cod < dist_red)  )
      called = 1; % 1 = cod;
   elseif ( (dist_capelin < dist_cod) & (dist_capelin < dist_red))
      called = 2; % 2 = capelin
   elseif ((dist_red < dist_cod) & (dist_red < dist_capelin) )
      called = 3; % 3 = redfish
   else
      called = 4; % we have a tie
      printf('\nwe have a tie with the distances\n');
      pause;
   end

   if called == 1
      if (i <= num_cod)
         cod_called_cod = cod_called_cod + 1;
      elseif (i > num_cod) & (i <= (num_cod + num_capelin))
         capelin_called_cod = capelin_called_cod + 1;
      else
         redfish_called_cod = redfish_called_cod + 1;
      end
   elseif called == 2
      if (i <= num_cod)
         cod_called_cap = cod_called_cap + 1;
      elseif (i > num_cod) & (i <= (num_cod + num_capelin))
         capelin_called_cap = capelin_called_cap + 1;
      else
         redfish_called_cap = redfish_called_cap + 1;
      end
   elseif called == 3
      if (i <= num_cod)
         cod_called_red = cod_called_red + 1;
      elseif (i > num_cod) & (i <= (num_cod + num_capelin))
         capelin_called_red = capelin_called_red + 1;
      else
         redfish_called_red = redfish_called_red + 1;
      end
   end

end % end of i = 1:length(alldata)
```

```
fprintf('Number of Features = %d\nFeature List = ',length(keep1));
fprintf('%d  ', keep1);
fprintf('\n\t\t\t\tPREDICTED CLASS\n');
fprintf('\t\tCOD\t\tCAPELIN\t\tREDFISH\n');

fprintf('COD\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n',...
   cod_called_cod,cod_called_cod/num_cod*100,...
   cod_called_cap,cod_called_cap/num_cod*100,...
   cod_called_red,cod_called_red/num_cod*100);

fprintf('CAPELIN\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n',...
   capelin_called_cod,capelin_called_cod/num_capelin*100,...
   capelin_called_cap,capelin_called_cap/num_capelin*100,...
   capelin_called_red,capelin_called_red/num_capelin*100);

fprintf('REDFISH\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n\n',...
   redfish_called_cod,redfish_called_cod/num_redfish*100,...
   redfish_called_cap,redfish_called_cap/num_redfish*100,...
   redfish_called_red,redfish_called_red/num_redfish*100);


overall_performance = (ccd_called_cod/num_cod*100 +
capelin_called_cap/num_capelin*100 + redfish_called_red/num_redfish*100)/3


% to illustrate over training

perf(17) = 86.4285;%, num_feats = 17
perf(16) = 88.1698;%, num_feats = 16
perf(15) = 89.4136;%, num_feats = 15
perf(14) = 89.2187;%, num_feats = 14
perf(13) = 90.1136;%, num_feats = 13
perf(12) = 88.9225;%, num_feats = 12
perf(11) = 91.1075;%, num_feats = 11
perf(10) = 91.0700;%, num_feats = 10
perf(9) = 91.7150;%, num_feats = 9
perf(8) = 91.7150;%, num_feats = 8
perf(7) = 91.7239;%, num_feats = 7
perf(6) = 90.2852;%, num_feats = 6
perf(5) = 89.4465;%, num_feats = 5
perf(4) = 87.7216;%, num_feats = 4
perf(3) = 84.1180;%num_feats = 3
perf(2) = 81.6040;%, num_feats = 2
perf(1) = 67.0804;%, num_feats = 1

plot(perf); grid; xlabel('number of features used'); ylabel('average classification
accuracy (percentage)');
```

```
%   classc6.m
%
%      Patricia LeFeuvre
%      Feb 25, 2001
%      Thesis classifier configuration #6
%      2 nodes:
%      node #1 - Mah - extract cod
%      node #2 - Mah - separate capelin and redfish

clear,
load capelin.txt;
load cod.txt;
load redfish.txt;

% Normalize
MN_red = min(redfish);
MN_capelin = min(capelin);
MN_cod = min(cod);
MN = min(min(MN_red,MN_capelin),MN_cod);

MX_red = max(redfish);
MX_capelin = max(capelin);
MX_cod = max(cod);
MX = max(max(MX_red,MX_capelin),MX_cod)-MN;

for i = 1:length(redfish)
      norm_red(i,1:25) = (redfish(i,1:25) - MN)./MX;
end

for i = 1:length(capelin)
      norm_capelin(i,1:25) = (capelin(i,1:25) - MN)./MX;
end

for i = 1:length(cod)
      norm_cod(i,1:25) = (cod(i,1:25) - MN)./MX;
end

num_cod = length(norm_cod);
num_capelin = length(norm_capelin);
num_redfish = length(norm_red);

% Classifier #6
alldata = [norm_cod; norm_capelin; norm_red];


% list of features selected
% Node 1 - extract cod
keep1 = [2 3 4 6 7 11 12 15];
%Node 2 - capelin vs. redfish
keep2 = [4 7 10 15];

cod_called_red = 0;
cod_called_cod = 0;
cod_called_cap = 0;
capelin_called_red = 0;
capelin_called_cod = 0;
capelin_called_cap = 0;
redfish_called_red = 0;
```

```
redfish_called_cod = 0;
redfish_called_cap = 0;

for i = 1:length(alldata)

   M_cod1 = mean(norm_cod(:,keep1));
   M_capelin1 = mean(norm_capelin(:,keep1));
   M_red1 = mean(norm_red(:,keep1));

   K_cod1 = cov(norm_cod(:,keep1));
   K_capelin1 = cov(norm_capelin(:,keep1));
   K_red1 = cov(norm_red(:,keep1));

   M_cod2 = mean(norm_cod(:,keep2));
   M_capelin2 = mean(norm_capelin(:,keep2));
   M_red2 = mean(norm_red(:,keep2));

   K_cod2 = cov(norm_cod(:,keep2));
   K_capelin2 = cov(norm_capelin(:,keep2));
   K_red2 = cov(norm_red(:,keep2));

   % leave out the sample we are testing
   if( i<=num_cod)
      M_cod1 = mean(norm_cod(find(1:num_cod~=i),keep1));
      K_cod1 = cov(norm_cod(find(1:num_cod~=i),keep1));
      M_cod2 = mean(norm_cod(find(1:num_cod~=i),keep2));
      K_cod2 = cov(norm_cod(find(1:num_cod~=i),keep2));
   elseif (i > num_cod) & (i <= (num_cod + num_capelin))
      M_capelin1 = mean(norm_capelin(find(1:num_capelin~=(i-num_cod)),keep1));
      K_capelin1 = cov(norm_capelin(find(1:num_capelin~=(i-num_cod)),keep1));
      M_capelin2 = mean(norm_capelin(find(1:num_capelin~=(i-num_cod)),keep2));
      K_capelin2 = cov(norm_capelin(find(1:num_capelin~=(i-num_cod)),keep2));
   else
      M_red1 = mean(norm_red(find(1:num_redfish~=(i-num_cod-num_capelin)),keep1));
      K_red1 = cov(norm_red(find(1:num_redfish~=(i-num_cod-num_capelin)),keep1));
      M_red2 = mean(norm_red(find(1:num_redfish~=(i-num_cod-num_capelin)),keep2));
      K_red2 = cov(norm_red(find(1:num_redfish~=(i-num_cod-num_capelin)),keep2));
   end

   data = [alldata(i,keep1)];
   dist_cod = (data - M_cod1) * inv(K_cod1) * (data - M_cod1)';
   dist_capelin = (data - M_capelin1) * inv(K_capelin1) * (data - M_capelin1)';
   dist_red = (data - M_red1) * inv(K_red1) * (data - M_red1)';
   if ( (dist_cod < dist_capelin) & (dist_cod < dist_red)  )
    called = 1; % 1 = cod;
   else
      % Node 2
      data = [alldata(i,keep2)];
      dist_capelin = (data - M_capelin2) * inv(K_capelin2) * (data - M_capelin2)';
      dist_red = (data - M_red2) * inv(K_red2) * (data - M_red2)';

            if (dist_capelin < dist_red)
            called = 2; % 2 = capelin
      else
         called = 3; % 3 = redfish
      end
   end
```

```
   if called == 1
      if (i <= num_cod)
         cod_called_cod = cod_called_cod + 1;
      elseif (i > num_cod) & (i <= (num_cod + num_capelin))
         capelin_called_cod = capelin_called_cod + 1;
      else
         redfish_called_cod = redfish_called_cod + 1;
      end
   elseif called == 2
      if (i <= num_cod)
         cod_called_cap = cod_called_cap + 1;
      elseif (i > num_cod) & (i <= (num_cod + num_capelin))
         capelin_called_cap = capelin_called_cap + 1;
      else
         redfish_called_cap = redfish_called_cap + 1;
      end
   elseif called == 3
      if (i <= num_cod)
         cod_called_red = cod_called_red + 1;
      elseif (i > num_cod) & (i <= (num_cod + num_capelin))
         capelin_called_red = capelin_called_red + 1;
      else
         redfish_called_red = redfish_called_red + 1;
      end
   end

end % end of i = 1:length(alldata)


fprintf('Node #1 Number of Features = %d\nFeature List = ',length(keep1));
fprintf('%d  ', keep1);

fprintf('\nNode #2 Number of Features = %d\nFeature List = ',length(keep2));
fprintf('%d  ', keep2);

fprintf('\n\t\t\t\tPREDICTED CLASS\n');
fprintf('\t\tCOD\t\tCAPELIN\t\tREDFISH\n');

fprintf('COD\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n',...
   cod_called_cod,cod_called_cod/num_cod*100,...
   cod_called_cap,cod_called_cap/num_cod*100,...
   cod_called_red,cod_called_red/num_cod*100);

fprintf('CAPELIN\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n',...
   capelin_called_cod,capelin_called_cod/num_capelin*100,...
   capelin_called_cap,capelin_called_cap/num_capelin*100,...
   capelin_called_red,capelin_called_red/num_capelin*100);

fprintf('REDFISH\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n\n',...
   redfish_called_cod,redfish_called_cod/num_redfish*100,...
   redfish_called_cap,redfish_called_cap/num_redfish*100,...
   redfish_called_red,redfish_called_red/num_redfish*100);
```

```
%     classc7.m
%
%     Patricia LeFeuvre
%     Feb 26, 2001
%     Thesis classifier configuration #7
%     2 nodes:
%     node #1 - Mah - extract capelin
%     node #2 - Mah - separate cod and redfish

clear,
load capelin.txt;
load cod.txt;
load redfish.txt;

% Normalize
MN_red = min(redfish);
MN_capelin = min(capelin);
MN_cod = min(cod);
MN = min(min(MN_red,MN_capelin),MN_cod);

MX_red = max(redfish);
MX_capelin = max(capelin);
MX_cod = max(cod);
MX = max(max(MX_red,MX_capelin),MX_cod)-MN;

for i = 1:length(redfish)
      norm_red(i,1:25) = (redfish(i,1:25) - MN)./MX;
end

for i = 1:length(capelin)
      norm_capelin(i,1:25) = (capelin(i,1:25) - MN)./MX;
end

for i = 1:length(cod)
      norm_cod(i,1:25) = (cod(i,1:25) - MN)./MX;
end

num_cod = length(norm_cod);
num_capelin = length(norm_capelin);
num_redfish = length(norm_red);

% Classifier #7
alldata = [norm_cod; norm_capelin; norm_red];

% list of features selected

% Node 1 - extract capelin
keep1 = [3 4 7 11 15 16];

%Node 2 - cod vs. redfish
keep2 = [2 3 4 6 7 11 12 15];


cod_called_red = 0;
cod_called_cod = 0;
cod_called_cap = 0;
capelin_called_red = 0;
capelin_called_cod = 0;
```

```
capelin_called_cap = 0;
redfish_called_red = 0;
redfish_called_cod = 0;
redfish_called_cap = 0;

for i = 1:length(alldata)

   M_cod1 = mean(norm_cod(:,keep1));
   M_capelin1 = mean(norm_capelin(:,keep1));
   M_red1 = mean(norm_red(:,keep1));

   K_cod1 = cov(norm_cod(:,keep1));
   K_capelin1 = cov(norm_capelin(:,keep1));
   K_red1 = cov(norm_red(:,keep1));

   M_cod2 = mean(norm_cod(:,keep2));
   M_capelin2 = mean(norm_capelin(:,keep2));
   M_red2 = mean(norm_red(:,keep2));

   K_cod2 = cov(norm_cod(:,keep2));
   K_capelin2 = cov(norm_capelin(:,keep2));
   K_red2 = cov(norm_red(:,keep2));

 % leave out the sample we are testing
    if( i<=num_cod)
      M_cod1 = mean(norm_cod(find(1:num_cod~=i),keep1));
      K_cod1 = cov(norm_cod(find(1:num_cod~=i),keep1));
      M_cod2 = mean(norm_cod(find(1:num_cod~=i),keep2));
      K_cod2 = cov(norm_cod(find(1:num_cod~=i),keep2));
   elseif (i > num_cod) & (i <= (num_cod + num_capelin))
      M_capelin1 = mean(norm_capelin(find(1:num_capelin~=(i-num_cod)),keep1));
      K_capelin1 = cov(norm_capelin(find(1:num_capelin~=(i-num_cod)),keep1));
      M_capelin2 = mean(norm_capelin(find(1:num_capelin~=(i-num_cod)),keep2));
      K_capelin2 = cov(norm_capelin(find(1:num_capelin~=(i-num_cod)),keep2));
   else
      M_red1 = mean(norm_red(find(1:num_redfish~=(i-num_cod-num_capelin)),keep1));
      K_red1 = cov(norm_red(find(1:num_redfish~=(i-num_cod-num_capelin)),keep1));
      M_red2 = mean(norm_red(find(1:num_redfish~=(i-num_cod-num_capelin)),keep2));
      K_red2 = cov(norm_red(find(1:num_redfish~=(i-num_cod-num_capelin)),keep2));
   end

   data = [alldata(i,keep1)];
   dist_cod = (data - M_cod1) * inv(K_cod1) * (data - M_cod1)';
   dist_capelin = (data - M_capelin1) * inv(K_capelin1) * (data - M_capelin1)';
   dist_red = (data - M_red1) * inv(K_red1) * (data - M_red1)';

   if ( (dist_capelin < dist_cod) & (dist_capelin < dist_red))
        called = 2; % 2 = capelin;
   else
        % Node 2
        data = [alldata(i,keep2)];
          dist_red = (data - M_red2) * inv(K_red2) * (data - M_red2)';
      dist_cod = (data - M_cod2) * inv(K_cod2) * (data - M_cod2)';

            if (dist_cod <= dist_red)
                called = 1; % 1 = cod
        else
            called = 3; % 3 = redfish
```

```
        end
    end

    if called == 1
        if (i <= num_cod)
            cod_called_cod = cod_called_cod + 1;
        elseif (i > num_cod) & (i <= (num_cod + num_capelin))
            capelin_called_cod = capelin_called_cod + 1;
        else
            redfish_called_cod = redfish_called_cod + 1;
        end
    elseif called == 2
        if (i <= num_cod)
            cod_called_cap = cod_called_cap + 1;
        elseif (i > num_cod) & (i <= (num_cod + num_capelin))
            capelin_called_cap = capelin_called_cap + 1;
        else
            redfish_called_cap = redfish_called_cap + 1;
        end
    elseif called == 3
        if (i <= num_cod)
            cod_called_red = cod_called_red + 1;
        elseif (i > num_cod) & (i <= (num_cod + num_capelin))
            capelin_called_red = capelin_called_red + 1;
        else
            redfish_called_red = redfish_called_red + 1;
        end
    end

end % end of i = 1:length(alldata)


fprintf('Number of Features Node 1 = %d\nFeature List = ',length(keep1));
fprintf('%d  ', keep1);
fprintf('\nNumber of Features Node 2= %d\nFeature List = ',length(keep2));
fprintf('%d  ', keep2);
fprintf('\n\t\t\t\tPREDICTED CLASS\n');
fprintf('\t\tCOD\t\tCAPELIN\t\tREDFISH\n');

fprintf('COD\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n',...
    cod_called_cod,cod_called_cod/num_cod*100,...
    cod_called_cap,cod_called_cap/num_cod*100,...
    cod_called_red,cod_called_red/num_cod*100);

fprintf('CAPELIN\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n',...
    capelin_called_cod,capelin_called_cod/num_capelin*100,...
    capelin_called_cap,capelin_called_cap/num_capelin*100,...
    capelin_called_red,capelin_called_red/num_capelin*100);

fprintf('REDFISH\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n\n',...
    redfish_called_cod,redfish_called_cod/num_redfish*100,...
    redfish_called_cap,redfish_called_cap/num_redfish*100,...
    redfish_called_red,redfish_called_red/num_redfish*100);
```

```
% classc8.m
%
%     Patricia LeFeuvre
%     Feb 26, 2001
%   Thesis classifier configuration #8
%   2 nodes:
%   node #1 - MAH - extract redfish
%   node #2 - MAH - separate cod and capelin

clear,
load capelin.txt;
load cod.txt;
load redfish.txt;

% Normalize
MN_red = min(redfish);
MN_capelin = min(capelin);
MN_cod = min(cod);
MN = min(min(MN_red,MN_capelin),MN_cod);

MX_red = max(redfish);
MX_capelin = max(capelin);
MX_cod = max(cod);
MX = max(max(MX_red,MX_capelin),MX_cod)-MN;

for i = 1:length(redfish)
      norm_red(i,1:25) = (redfish(i,1:25) - MN)./MX;
end

for i = 1:length(capelin)
      norm_capelin(i,1:25) = (capelin(i,1:25) - MN)./MX;
end

for i = 1:length(cod)
      norm_cod(i,1:25) = (cod(i,1:25) - MN)./MX;
end

num_cod = length(norm_cod);
num_capelin = length(norm_capelin);
num_redfish = length(norm_red);

% Classifier #8
alldata = [norm_cod; norm_capelin; norm_red];
node2data = [norm_cod; norm_capelin];

% list of features selected
% Node 1 - extract redfish
keep1 = [2 3 4 6 7 11 12 15 16];
%Node 2 - cod vs. capelin
keep2 = [2  3  6  11  12  15  16  20  22  25];

training_cod = norm_cod;
training_capelin = norm_capelin;
training_red = norm_red;

cod_called_red = 0;
cod_called_cod = 0;
cod_called_cap = 0;
```

```
capelin_called_red = 0;
capelin_called_cod = 0;
capelin_called_cap = 0;
redfish_called_red = 0;
redfish_called_cod = 0;
redfish_called_cap = 0;

for i = 1:length(alldata)
   M_cod1 = mean(training_cod(:,keep1));
   M_capelin1 = mean(training_capelin(:,keep1));
   M_red1 = mean(training_red(:,keep1));
   K_cod1 = cov(training_cod(:,keep1));
   K_capelin1 = cov(training_capelin(:,keep1));
   K_red1 = cov(training_red(:,keep1));
   M_cod2 = mean(training_cod(:,keep2));
   M_capelin2 = mean(training_capelin(:,keep2));
   M_red2 = mean(training_red(:,keep2));
   K_cod2 = cov(training_cod(:,keep2));
   K_capelin2 = cov(training_capelin(:,keep2));
   K_red2 = cov(training_red(:,keep2));

 % leave out the sample we are testing
 %if(1==2)
   if( i<=num_cod)
      M_cod1 = mean(training_cod(find(1:num_cod~=i),keep1));
      K_cod1 = cov(training_cod(find(1:num_cod~=i),keep1));
      M_cod2 = mean(training_cod(find(1:num_cod~=i),keep2));
      K_cod2 = cov(training_cod(find(1:num_cod~=i),keep2));
   elseif (i > num_cod) & (i <= (num_cod + num_capelin))
      M_capelin1 = mean(training_capelin(find(1:num_capelin~=(i-num_cod)),keep1));
      K_capelin1 = cov(training_capelin(find(1:num_capelin~=(i-num_cod)),keep1));

      M_capelin2 = mean(training_capelin(find(1:num_capelin~=(i-num_cod)),keep2));
      K_capelin2 = cov(training_capelin(find(1:num_capelin~=(i-num_cod)),keep2));

   else
      M_red1 =
mean(training_red(find(1:num_redfish~=(i-num_cod-num_capelin)),keep1));
      K_red1 =
cov(training_red(find(1:num_redfish~=(i-num_cod-num_capelin)),keep1));
      M_red2 =
mean(training_red(find(1:num_redfish~=(i-num_cod-num_capelin)),keep2));
      K_red2 =
cov(training_red(find(1:num_redfish~=(i-num_cod-num_capelin)),keep2));
   end
 %end

   data = [alldata(i,keep1)];
   dist_cod = (data - M_cod1) * inv(K_cod1) * (data - M_cod1)';
   dist_capelin = (data - M_capelin1) * inv(K_capelin1) * (data - M_capelin1)';
   dist_red = (data - M_red1) * inv(K_red1) * (data - M_red1)';
   if ( (dist_red < dist_cod) & (dist_red< dist_capelin))
        called = 3; % 3 = redfish;
   else
      % Node 2
        data = [alldata(i,keep2)];
          dist_capelin = (data - M_capelin2) * inv(K_capelin2) * (data -
M_capelin2)';
```

```
      dist_cod = (data - M_cod2) * inv(K_cod2) * (data - M_cod2)';

            if (dist_cod <= dist_capelin)
                called = 1; % 1 = cod
        else
            called = 2; % 2 = capelin
        end
    end

    if called == 1
       if (i <= num_cod)
          cod_called_cod = cod_called_cod + 1;
       elseif (i > num_cod) & (i <= (num_cod + num_capelin))
          capelin_called_cod = capelin_called_cod + 1;
       else
          redfish_called_cod = redfish_called_cod + 1;
       end
    elseif called == 2
       if (i <= num_cod)
          cod_called_cap = cod_called_cap + 1;
       elseif (i > num_cod) & (i <= (num_cod + num_capelin))
          capelin_called_cap = capelin_called_cap + 1;
       else
          redfish_called_cap = redfish_called_cap + 1;
       end
    elseif called == 3
       if (i <= num_cod)
          cod_called_red = cod_called_red + 1;
       elseif (i > num_cod) & (i <= (num_cod + num_capelin))
          capelin_called_red = capelin_called_red + 1;
       else
          redfish_called_red = redfish_called_red + 1;
       end
    end

end % end of i = 1:length(alldata)


fprintf('Number of Features Node 1 = %d\nFeature List = ',length(keep1));
fprintf('%d  ', keep1);
fprintf('\nNumber of Features Node 2= %d\nFeature List = ',length(keep2));
fprintf('%d  ', keep2);
fprintf('\n\t\t\t\tPREDICTED CLASS\n');
fprintf('\t\tCOD\t\tCAPELIN\t\tREDFISH\n');

fprintf('COD\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n',...
   cod_called_cod,cod_called_cod/num_cod*100,...
   cod_called_cap,cod_called_cap/num_cod*100,...
   cod_called_red,cod_called_red/num_cod*100);
fprintf('CAPELIN\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n',...
   capelin_called_cod,capelin_called_cod/num_capelin*100,...
   capelin_called_cap,capelin_called_cap/num_capelin*100,...
   capelin_called_red,capelin_called_red/num_capelin*100);
fprintf('REDFISH\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n\n',...
   redfish_called_cod,redfish_called_cod/num_redfish*100,...
   redfish_called_cap,redfish_called_cap/num_redfish*100,...
   redfish_called_red,redfish_called_red/num_redfish*100);
```

```
%     classC9.m
%
%
%     Patricia LeFeuvre
%     April 18, 2002
%     Thesis classifier configuration #9
%     2 nodes:
%     node #1 - 3NN - extract cod [based on node 1 class config #2]
%     node #2 - Mah - separate capelin and redfish [based on node 2 class config
      #6]

clear,
load capelin.txt;
load cod.txt;
load redfish.txt;

%normalize all of the data
MN_red = min(redfish);
MN_capelin = min(capelin);
MN_cod = min(cod);
MN = min(min(MN_red,MN_capelin),MN_cod);
MX_red = max(redfish);
MX_capelin = max(capelin);
MX_cod = max(cod);
MX = max(max(MX_red,MX_capelin),MX_cod)-MN;

for i = 1:length(redfish)
      norm_red(i,1:25) = (redfish(i,1:25) - MN)./MX;
end

for i = 1:length(capelin)
      norm_capelin(i,1:25) = (capelin(i,1:25) - MN)./MX;
end

for i = 1:length(cod)
      norm_cod(i,1:25) = (cod(i,1:25) - MN)./MX;
end

alldata = [norm_cod; norm_capelin; norm_red];

cod_called_red = 0;
cod_called_cod = 0;
cod_called_cap = 0;
capelin_called_red = 0;
capelin_called_cod = 0;
capelin_called_cap = 0;
redfish_called_red = 0;
redfish_called_cod = 0;
redfish_called_cap = 0;

num_cod = length(norm_cod);
num_capelin = length(norm_capelin);
num_redfish = length(norm_red);


% list of features selected
% Node 1 - extract cod
keep1 =    [3 4 6 9 11 12 15 22 24 25];
```

```
% Node 2 - capelin vs. redfish
keep2 =    [4 7 10 15];


for i = 1:length(alldata)

   M_cod2 = mean(norm_cod(:,keep2));
   M_capelin2 = mean(norm_capelin(:,keep2));
   M_red2 = mean(norm_red(:,keep2));

   K_cod2 = cov(norm_cod(:,keep2));
   K_capelin2 = cov(norm_capelin(:,keep2));
   K_red2 = cov(norm_red(:,keep2));

   % leave out the sample we are testing
   if( i<=num_cod)
      M_cod2 = mean(norm_cod(find(1:num_cod~=i),keep2));
      K_cod2 = cov(norm_cod(find(1:num_cod~=i),keep2));
   elseif (i > num_cod) & (i <= (num_cod + num_capelin))
      M_capelin2 = mean(norm_capelin(find(1:num_capelin~=(i-num_cod)),keep2));
      K_capelin2 = cov(norm_capelin(find(1:num_capelin~=(i-num_cod)),keep2));
   else
      M_red2 = mean(norm_red(find(1:num_redfish~=(i-num_cod-num_capelin)),keep2));
      K_red2 = cov(norm_red(find(1:num_redfish~=(i-num_cod-num_capelin)),keep2));
   end


      data = [alldata(i,keep1)];
      for j = 1:length(alldata)
            if i ~= j
                  tmp = data - alldata(j,keep1);
                  dist(j) = sqrt(tmp*tmp');
            else
                  dist(j) = 25;
            end
      end

      [sortedDist, Index] = sort(dist);

      if ( length(find(Index(1:3) <= num_cod )) >= 2) % 2/3 were cod
         called = 1; % 1 = cod;
    else
      % Node 2
        data = [alldata(i,keep2)];
      dist_capelin = (data - M_capelin2) * inv(K_capelin2) * (data - M_capelin2)';
          dist_red = (data - M_red2) * inv(K_red2) * (data - M_red2)';

            if (dist_capelin < dist_red)
                called = 2; % 2 = capelin
         else
            called = 3; % 3 = redfish
        end
    end  % end of Node 2

      if called == 1
            if (i <= num_cod)
                  cod_called_cod = cod_called_cod + 1;
            elseif (i > num_cod) & (i <= (num_cod + num_capelin))
```

```
                capelin_called_cod = capelin_called_cod + 1;
            else
                redfish_called_cod = redfish_called_cod + 1;
        end
    elseif called == 2
            if (i <= num_cod)
                cod_called_cap = cod_called_cap + 1;
            elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                capelin_called_cap = capelin_called_cap + 1;
            else
                redfish_called_cap = redfish_called_cap + 1;
            end
    else
            if (i <= num_cod)
                cod_called_red = cod_called_red + 1;
            elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                capelin_called_red = capelin_called_red + 1;
            else
                redfish_called_red = redfish_called_red + 1;
            end
   end
end


fprintf('Node #1 Number of Features = %d\nFeature List = ',length(keep1));
fprintf('%d  ', keep1);

fprintf('\nNode #2 Number of Features = %d\nFeature List = ',length(keep2));
fprintf('%d  ', keep2);

fprintf('\n\t\t\t\tPREDICTED CLASS\n');
fprintf('\t\tCOD\t\tCAPELIN\t\tREDFISH\n');

fprintf('COD\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n',...
cod_called_cod,cod_called_cod/num_cod*100,...
cod_called_cap,cod_called_cap/num_cod*100,...
cod_called_red,cod_called_red/num_cod*100);

fprintf('CAPELIN\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n',...
capelin_called_cod,capelin_called_cod/num_capelin*100,...
capelin_called_cap,capelin_called_cap/num_capelin*100,...
capelin_called_red,capelin_called_red/num_capelin*100);

fprintf('REDFISH\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n\n',...
redfish_called_cod,redfish_called_cod/num_redfish*100,...
redfish_called_cap,redfish_called_cap/num_redfish*100,...
redfish_called_red,redfish_called_red/num_redfish*100);
```

```
%   classC10.m
%
%   Patricia LeFeuvre
%   April 18, 2002
%   Thesis classifier configuration #10
%   2 nodes:
%   node #1 - 3NN - extract capelin [based on node 1 class config #3]
%   node #2 - Mah - separate cod and redfish [based on node 2 class config #7]

clear,
load capelin.txt;
load cod.txt;
load redfish.txt;

%normalize all of the data
MN_red = min(redfish);
MN_capelin = min(capelin);
MN_cod = min(cod);
MN = min(min(MN_red,MN_capelin),MN_cod);

MX_red = max(redfish);
MX_capelin = max(capelin);
MX_cod = max(cod);
MX = max(max(MX_red,MX_capelin),MX_cod)-MN;

for i = 1:length(redfish)
      norm_red(i,1:25) = (redfish(i,1:25) - MN)./MX;
end

for i = 1:length(capelin)
      norm_capelin(i,1:25) = (capelin(i,1:25) - MN)./MX;
end

for i = 1:length(cod)
      norm_cod(i,1:25) = (cod(i,1:25) - MN)./MX;
end

alldata = [norm_cod; norm_capelin; norm_red];

cod_called_red = 0;
cod_called_cod = 0;
cod_called_cap = 0;
capelin_called_red = 0;
capelin_called_cod = 0;
capelin_called_cap = 0;
redfish_called_red = 0;
redfish_called_cod = 0;
redfish_called_cap = 0;

num_cod = length(norm_cod);
num_capelin = length(norm_capelin);
num_redfish = length(norm_red);


% list of features selected
% Node 1 - extract capelin
keep1 =    [3 11 15 22 25];
```

```
% Node 2 - cod vs. redfish
keep2 =    [2 3 4 6 7 11 12 15];


for i = 1:length(alldata)

   M_cod2 = mean(norm_cod(:,keep2));
   M_capelin2 = mean(norm_capelin(:,keep2));
   M_red2 = mean(norm_red(:,keep2));

   K_cod2 = cov(norm_cod(:,keep2));
   K_capelin2 = cov(norm_capelin(:,keep2));
   K_red2 = cov(norm_red(:,keep2));

   % leave out the sample we are testing
   if( i<=num_cod)
      M_cod2 = mean(norm_cod(find(1:num_cod~=i),keep2));
      K_cod2 = cov(norm_cod(find(1:num_cod~=i),keep2));
   elseif (i > num_cod) & (i <= (num_cod + num_capelin))
      M_capelin2 = mean(norm_capelin(find(1:num_capelin~=(i-num_cod)),keep2));
      K_capelin2 = cov(norm_capelin(find(1:num_capelin~=(i-num_cod)),keep2));
   else
      M_red2 = mean(norm_red(find(1:num_redfish~=(i-num_cod-num_capelin)),keep2));
      K_red2 = cov(norm_red(find(1:num_redfish~=(i-num_cod-num_capelin)),keep2));
   end

      data = [alldata(i,keep1)];
      for j = 1:length(alldata)
            if i ~= j
                  tmp = data - alldata(j,keep1);
                  dist(j) = sqrt(tmp*tmp');
            else
                  dist(j) = 25;
            end
      end

      [sortedDist, Index] = sort(dist);

      if ( length(find( (Index(1:3) > num_cod) & (Index(1:3) <=
(num_cod+num_capelin))) ) >= 2) % 2/3 were capelin
          called = 2; % 2 = capelin;
    else
      % Node 2
        data = [alldata(i,keep2)];
      dist_cod= (data - M_cod2) * inv(K_cod2) * (data - M_cod2)';
          dist_red = (data - M_red2) * inv(K_red2) * (data - M_red2)';

            if (dist_cod < dist_red)
                called = 1; % 1 = cod
        else
            called = 3; % 3 = redfish
        end
    end  % end of Node 2

      if called == 1
            if (i <= num_cod)
                  cod_called_cod = cod_called_cod + 1;
            elseif (i > num_cod) & (i <= (num_cod + num_capelin))
```

```
                capelin_called_cod = capelin_called_cod + 1;
            else
                redfish_called_cod = redfish_called_cod + 1;
        end
    elseif called == 2
            if (i <= num_cod)
                cod_called_cap = cod_called_cap + 1;
            elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                capelin_called_cap = capelin_called_cap + 1;
            else
                redfish_called_cap = redfish_called_cap + 1;
            end
    else
            if (i <= num_cod)
                cod_called_red = cod_called_red + 1;
            elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                capelin_called_red = capelin_called_red + 1;
            else
                redfish_called_red = redfish_called_red + 1;
            end
   end
end


fprintf('Node #1 Number of Features = %d\nFeature List = ',length(keep1));
fprintf('%d  ', keep1);

fprintf('\nNode #2 Number of Features = %d\nFeature List = ',length(keep2));
fprintf('%d  ', keep2);

fprintf('\n\t\t\t\tPREDICTED CLASS\n');
fprintf('\t\tCOD\t\tCAPELIN\t\tREDFISH\n');

fprintf('COD\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n',...
cod_called_cod,cod_called_cod/num_cod*100,...
cod_called_cap,cod_called_cap/num_cod*100,...
cod_called_red,cod_called_red/num_cod*100);

fprintf('CAPELIN\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n',...
capelin_called_cod,capelin_called_cod/num_capelin*100,...
capelin_called_cap,capelin_called_cap/num_capelin*100,...
capelin_called_red,capelin_called_red/num_capelin*100);

fprintf('REDFISH\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n\n',...
redfish_called_cod,redfish_called_cod/num_redfish*100,...
redfish_called_cap,redfish_called_cap/num_redfish*100,...
redfish_called_red,redfish_called_red/num_redfish*100);
```

```
%   classC11.m
%
%   Patricia LeFeuvre
%   April 23, 2002
%   Thesis classifier configuration #11
%   2 nodes:
%   node #1 - 3NN - extract redfish [based on node 1 class config #4]
%   node #2 - Mah - separate cod and capelin [based on node 2 class config #8]

clear,
load capelin.txt;
load cod.txt;
load redfish.txt;

%normalize all of the data
MN_red = min(redfish);
MN_capelin = min(capelin);
MN_cod = min(cod);
MN = min(min(MN_red,MN_capelin),MN_cod);

MX_red = max(redfish);
MX_capelin = max(capelin);
MX_cod = max(cod);
MX = max(max(MX_red,MX_capelin),MX_cod)-MN;

for i = 1:length(redfish)
      norm_red(i,1:25) = (redfish(i,1:25) - MN)./MX;
end

for i = 1:length(capelin)
      norm_capelin(i,1:25) = (capelin(i,1:25) - MN)./MX;
end

for i = 1:length(cod)
      norm_cod(i,1:25) = (cod(i,1:25) - MN)./MX;
end

alldata = [norm_cod; norm_capelin; norm_red];

cod_called_red = 0;
cod_called_cod = 0;
cod_called_cap = 0;
capelin_called_red = 0;
capelin_called_cod = 0;
capelin_called_cap = 0;
redfish_called_red = 0;
redfish_called_cod = 0;
redfish_called_cap = 0;

num_cod = length(norm_cod);
num_capelin = length(norm_capelin);
num_redfish = length(norm_red);


% list of features selected
% Node 1 - extract redfish
keep1 =    [4 9 10 11 12 15 16 20 24 25];
```

```
% Node 2 - cod vs. capelin
keep2 = [2  3  6  11  12  15  16  20  22  25];


for i = 1:length(alldata)

   M_cod2 = mean(norm_cod(:,keep2));
   M_capelin2 = mean(norm_capelin(:,keep2));
   M_red2 = mean(norm_red(:,keep2));

   K_cod2 = cov(norm_cod(:,keep2));
   K_capelin2 = cov(norm_capelin(:,keep2));
   K_red2 = cov(norm_red(:,keep2));

   % leave out the sample we are testing
   if( i<=num_cod)
      M_cod2 = mean(norm_cod(find(1:num_cod~=i),keep2));
      K_cod2 = cov(norm_cod(find(1:num_cod~=i),keep2));
   elseif (i > num_cod) & (i <= (num_cod + num_capelin))
      M_capelin2 = mean(norm_capelin(find(1:num_capelin~=(i-num_cod)),keep2));
      K_capelin2 = cov(norm_capelin(find(1:num_capelin~=(i-num_cod)),keep2));
   else
      M_red2 = mean(norm_red(find(1:num_redfish~=(i-num_cod-num_capelin)),keep2));
      K_red2 = cov(norm_red(find(1:num_redfish~=(i-num_cod-num_capelin)),keep2));
   end


      data = [alldata(i,keep1)];
      for j = 1:length(alldata)
            if i ~= j
                  tmp = data - alldata(j,keep1);
                  dist(j) = sqrt(tmp*tmp');
            else
                  dist(j) = 25;
            end
      end

      [sortedDist, Index] = sort(dist);

      if ( length(find(Index(1:3) > (num_cod+num_capelin))) >= 2)% at least 2/3
were redfish
          called = 3; % 3 = redfish;
    else
      % Node 2
        data = [alldata(i,keep2)];
          dist_capelin = (data - M_capelin2) * inv(K_capelin2) * (data -
M_capelin2)';
      dist_cod = (data - M_cod2) * inv(K_cod2) * (data - M_cod2)';

            if (dist_cod <= dist_capelin)
                called = 1; % 1 = cod
        else
            called = 2; % 2 = capelin
        end
    end


      if called == 1
```

```
            if (i <= num_cod)
                  cod_called_cod = cod_called_cod + 1;
            elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                  capelin_called_cod = capelin_called_cod + 1;
            else
                  redfish_called_cod = redfish_called_cod + 1;
        end
    elseif called == 2
            if (i <= num_cod)
                  cod_called_cap = cod_called_cap + 1;
            elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                  capelin_called_cap = capelin_called_cap + 1;
            else
                  redfish_called_cap = redfish_called_cap + 1;
            end
    else
            if (i <= num_cod)
                  cod_called_red = cod_called_red + 1;
            elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                  capelin_called_red = capelin_called_red + 1;
            else
                  redfish_called_red = redfish_called_red + 1;
            end
   end
end


fprintf('Node #1 Number of Features = %d\nFeature List = ',length(keep1));
fprintf('%d  ', keep1);

fprintf('\nNode #2 Number of Features = %d\nFeature List = ',length(keep2));
fprintf('%d  ', keep2);

fprintf('\n\t\t\t\tPREDICTED CLASS\n');
fprintf('\t\tCOD\t\tCAPELIN\t\tREDFISH\n');

fprintf('COD\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n',...
cod_called_cod,cod_called_cod/num_cod*100,...
cod_called_cap,cod_called_cap/num_cod*100,...
cod_called_red,cod_called_red/num_cod*100);

fprintf('CAPELIN\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n',...
capelin_called_cod,capelin_called_cod/num_capelin*100,...
capelin_called_cap,capelin_called_cap/num_capelin*100,...
capelin_called_red,capelin_called_red/num_capelin*100);

fprintf('REDFISH\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n\n',...
redfish_called_cod,redfish_called_cod/num_redfish*100,...
redfish_called_cap,redfish_called_cap/num_redfish*100,...
redfish_called_red,redfish_called_red/num_redfish*100);
```

```
%   classC12.m
%
%     Patricia LeFeuvre
%     April 23, 2002
%   Thesis classifier configuration #12
%   2 nodes:
%   node #1 - Mah - extract cod [based on node 1 class config #6]
%   node #2 - 3NN - separate capelin and redfish [based on node 2 class config #2]

clear,
load capelin.txt;
load cod.txt;
load redfish.txt;

%normalize all of the data
MN_red = min(redfish);
MN_capelin = min(capelin);
MN_cod = min(cod);
MN = min(min(MN_red,MN_capelin),MN_cod);

MX_red = max(redfish);
MX_capelin = max(capelin);
MX_cod = max(cod);
MX = max(max(MX_red,MX_capelin),MX_cod)-MN;

for i = 1:length(redfish)
      norm_red(i,1:25) = (redfish(i,1:25) - MN)./MX;
end

for i = 1:length(capelin)
      norm_capelin(i,1:25) = (capelin(i,1:25) - MN)./MX;
end

for i = 1:length(cod)
      norm_cod(i,1:25) = (cod(i,1:25) - MN)./MX;
end

alldata = [norm_cod; norm_capelin; norm_red];
node2data = [norm_capelin; norm_red];

cod_called_red = 0;
cod_called_cod = 0;
cod_called_cap = 0;
capelin_called_red = 0;
capelin_called_cod = 0;
capelin_called_cap = 0;
redfish_called_red = 0;
redfish_called_cod = 0;
redfish_called_cap = 0;

num_cod = length(norm_cod);
num_capelin = length(norm_capelin);
num_redfish = length(norm_red);


% list of features selected
% Node 1 - extract cod
keep1 = [2 3 4 6 7 11 12 15];
```

```
% Node 2 - capelin vs. redfish
keep2 =  [3 9 11 15 22 24 25];


for i = 1:length(alldata)

   M_cod1 = mean(norm_cod(:,keep1));
   M_capelin1 = mean(norm_capelin(:,keep1));
   M_red1 = mean(norm_red(:,keep1));

   K_cod1 = cov(norm_cod(:,keep1));
   K_capelin1 = cov(norm_capelin(:,keep1));
   K_red1 = cov(norm_red(:,keep1));

   % leave out the sample we are testing
   if( i<=num_cod)
      M_cod1 = mean(norm_cod(find(1:num_cod~=i),keep1));
      K_cod1 = cov(norm_cod(find(1:num_cod~=i),keep1));
   elseif (i > num_cod) & (i <= (num_cod + num_capelin))
      M_capelin1 = mean(norm_capelin(find(1:num_capelin~=(i-num_cod)),keep1));
      K_capelin1 = cov(norm_capelin(find(1:num_capelin~=(i-num_cod)),keep1));
   else
      M_red1 = mean(norm_red(find(1:num_redfish~=(i-num_cod-num_capelin)),keep1));
      K_red1 = cov(norm_red(find(1:num_redfish~=(i-num_cod-num_capelin)),keep1));
   end

    data = [alldata(i,keep1)];
    dist_cod = (data - M_cod1) * inv(K_cod1) * (data - M_cod1)';
    dist_capelin = (data - M_capelin1) * inv(K_capelin1) * (data - M_capelin1)';
    dist_red = (data - M_red1) * inv(K_red1) * (data - M_red1)';
    if ( (dist_cod < dist_capelin) & (dist_cod < dist_red)  )
        called = 1; % 1 = cod;
    else
      % Node 2
      data = [alldata(i,keep2)];
        sameCount = 0;
      for j = 1:length(node2data)
            tmp = data - node2data(j,keep2);
            dist(j) = sqrt(tmp*tmp');
            if (dist(j) == 0) % the same point so don't use
               dist(j) == 25;
               sameCount = sameCount + 1;
               if sameCount > 1
                  fprintf('Error - too many identical points ');
               end

            end
      end
      [sortedDist, Index] = sort(dist);

      if ( length(find(Index(1:3) <= num_capelin )) >= 2) % 2/3 were capelin
            called = 2; % 2 = capelin
      else
            called = 3; % 3 = redfish
      end

   end  % end of Node 2
```

```
      if called == 1
            if (i <= num_cod)
                  cod_called_cod = cod_called_cod + 1;
            elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                  capelin_called_cod = capelin_called_cod + 1;
            else
                  redfish_called_cod = redfish_called_cod + 1;
      end

      elseif called == 2
            if (i <= num_cod)
                  cod_called_cap = cod_called_cap + 1;
            elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                  capelin_called_cap = capelin_called_cap + 1;
            else
                  redfish_called_cap = redfish_called_cap + 1;
            end

      else
            if (i <= num_cod)
                  cod_called_red = cod_called_red + 1;
            elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                  capelin_called_red = capelin_called_red + 1;
            else
                  redfish_called_red = redfish_called_red + 1;
            end
   end

end

fprintf('Node #1 Number of Features = %d\nFeature List = ',length(keep1));
fprintf('%d  ', keep1);

fprintf('\nNode #2 Number of Features = %d\nFeature List = ',length(keep2));
fprintf('%d  ', keep2);

fprintf('\n\t\t\t\tPREDICTED CLASS\n');
fprintf('\t\tCOD\t\tCAPELIN\t\tREDFISH\n');

fprintf('COD\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n',...
cod_called_cod,cod_called_cod/num_cod*100,...
cod_called_cap,cod_called_cap/num_cod*100,...
cod_called_red,cod_called_red/num_cod*100);

fprintf('CAPELIN\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n',...
capelin_called_cod,capelin_called_cod/num_capelin*100,...
capelin_called_cap,capelin_called_cap/num_capelin*100,...
capelin_called_red,capelin_called_red/num_capelin*100);

fprintf('REDFISH\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n\n',...
redfish_called_cod,redfish_called_cod/num_redfish*100,...
redfish_called_cap,redfish_called_cap/num_redfish*100,...
redfish_called_red,redfish_called_red/num_redfish*100);
```

```
%   classC13.m
%
%     Patricia LeFeuvre
%     April 23, 2002
%   Thesis classifier configuration #13
%   2 nodes:
%   node #1 - Mah - extract capelin [based on node 1 class config #7]
%   node #2 - 3NN - separate cod and redfish [based on node 2 class config #3]

clear,
load capelin.txt;
load cod.txt;
load redfish.txt;

%normalize all of the data
MN_red = min(redfish);
MN_capelin = min(capelin);
MN_cod = min(cod);
MN = min(min(MN_red,MN_capelin),MN_cod);

MX_red = max(redfish);
MX_capelin = max(capelin);
MX_cod = max(cod);
MX = max(max(MX_red,MX_capelin),MX_cod)-MN;

for i = 1:length(redfish)
      norm_red(i,1:25) = (redfish(i,1:25) - MN)./MX;
end

for i = 1:length(capelin)
      norm_capelin(i,1:25) = (capelin(i,1:25) - MN)./MX;
end

for i = 1:length(cod)
      norm_cod(i,1:25) = (cod(i,1:25) - MN)./MX;
end

alldata = [norm_cod; norm_capelin; norm_red];
node2data = [norm_cod; norm_red];

cod_called_red = 0;
cod_called_cod = 0;
cod_called_cap = 0;
capelin_called_red = 0;
capelin_called_cod = 0;
capelin_called_cap = 0;
redfish_called_red = 0;
redfish_called_cod = 0;
redfish_called_cap = 0;

num_cod = length(norm_cod);
num_capelin = length(norm_capelin);
num_redfish = length(norm_red);


% list of features selected
% Node 1 - extract capelin
keep1 = [3 4 7 11 15 16];
```

```
% Node 2 - cod vs. redfish
keep2 =    [4 9 12 15 20];


for i = 1:length(alldata)

   M_cod1 = mean(norm_cod(:,keep1));
   M_capelin1 = mean(norm_capelin(:,keep1));
   M_red1 = mean(norm_red(:,keep1));

   K_cod1 = cov(norm_cod(:,keep1));
   K_capelin1 = cov(norm_capelin(:,keep1));
   K_red1 = cov(norm_red(:,keep1));

   % leave out the sample we are testing
   if( i<=num_cod)
      M_cod1 = mean(norm_cod(find(1:num_cod~=i),keep1));
      K_cod1 = cov(norm_cod(find(1:num_cod~=i),keep1));
   elseif (i > num_cod) & (i <= (num_cod + num_capelin))
      M_capelin1 = mean(norm_capelin(find(1:num_capelin~=(i-num_cod)),keep1));
      K_capelin1 = cov(norm_capelin(find(1:num_capelin~=(i-num_cod)),keep1));
   else
      M_red1 = mean(norm_red(find(1:num_redfish~=(i-num_cod-num_capelin)),keep1));
      K_red1 = cov(norm_red(find(1:num_redfish~=(i-num_cod-num_capelin)),keep1));
   end

    data = [alldata(i,keep1)];
    dist_cod = (data - M_cod1) * inv(K_cod1) * (data - M_cod1)';
    dist_capelin = (data - M_capelin1) * inv(K_capelin1) * (data - M_capelin1)';
    dist_red = (data - M_red1) * inv(K_red1) * (data - M_red1)';

    if ( (dist_capelin < dist_cod) & (dist_capelin < dist_red))
        called = 2; % 2 = capelin;
    else
      % Node 2
    % Node 2
        data = [alldata(i,keep2)];
          sameCount = 0;
        for j = 1:length(node2data)
                  tmp = data - node2data(j,keep2);
            dist2(j) = sqrt(tmp*tmp');
            if (dist2(j) == 0) % the same point so don't use
               dist2(j) == 25;
               sameCount = sameCount + 1;
               if sameCount > 1
                  fprintf('Error - too many identical points ');
               end
            end
        end
        [sortedDist, Index] = sort(dist2);

            if ( length(find(Index(1:3) <= num_cod )) >= 2) % 2/3 were cod
                  called = 1; % 1 = cod
          else
                called = 3; % 3 = redfish
        end

      end  % end of Node 2
```

```
      if called == 1
            if (i <= num_cod)
                  cod_called_cod = cod_called_cod + 1;
            elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                  capelin_called_cod = capelin_called_cod + 1;
            else
                  redfish_called_cod = redfish_called_cod + 1;
      end

      elseif called == 2
            if (i <= num_cod)
                  cod_called_cap = cod_called_cap + 1;
            elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                  capelin_called_cap = capelin_called_cap + 1;
            else
                  redfish_called_cap = redfish_called_cap + 1;
            end

      else
            if (i <= num_cod)
                  cod_called_red = cod_called_red + 1;
            elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                  capelin_called_red = capelin_called_red + 1;
            else
                  redfish_called_red = redfish_called_red + 1;
            end
   end

end

fprintf('Node #1 Number of Features = %d\nFeature List = ',length(keep1));
fprintf('%d  ', keep1);

fprintf('\nNode #2 Number of Features = %d\nFeature List = ',length(keep2));
fprintf('%d  ', keep2);

fprintf('\n\t\t\t\tPREDICTED CLASS\n');
fprintf('\t\tCOD\t\tCAPELIN\t\tREDFISH\n');

fprintf('COD\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n',...
cod_called_cod,cod_called_cod/num_cod*100,...
cod_called_cap,cod_called_cap/num_cod*100,...
cod_called_red,cod_called_red/num_cod*100);

fprintf('CAPELIN\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n',...
capelin_called_cod,capelin_called_cod/num_capelin*100,...
capelin_called_cap,capelin_called_cap/num_capelin*100,...
capelin_called_red,capelin_called_red/num_capelin*100);

fprintf('REDFISH\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n\n',...
redfish_called_cod,redfish_called_cod/num_redfish*100,...
redfish_called_cap,redfish_called_cap/num_redfish*100,...
redfish_called_red,redfish_called_red/num_redfish*100);
```

```
%   classC14.m
%
%     Patricia LeFeuvre
%     April 23, 2002
%   Thesis classifier configuration #14
%   2 nodes:
%   node #1 - Mah - extract capelin [based on node 1 class config #8]
%   node #2 - 3NN - separate cod and redfish [based on node 2 class config #4]

clear,
load capelin.txt;
load cod.txt;
load redfish.txt;

%normalize all of the data
MN_red = min(redfish);
MN_capelin = min(capelin);
MN_cod = min(cod);
MN = min(min(MN_red,MN_capelin),MN_cod);

MX_red = max(redfish);
MX_capelin = max(capelin);
MX_cod = max(cod);
MX = max(max(MX_red,MX_capelin),MX_cod)-MN;

for i = 1:length(redfish)
      norm_red(i,1:25) = (redfish(i,1:25) - MN)./MX;
end

for i = 1:length(capelin)
      norm_capelin(i,1:25) = (capelin(i,1:25) - MN)./MX;
end

for i = 1:length(cod)
      norm_cod(i,1:25) = (cod(i,1:25) - MN)./MX;
end

alldata = [norm_cod; norm_capelin; norm_red];
node2data = [norm_cod; norm_capelin];

cod_called_red = 0;
cod_called_cod = 0;
cod_called_cap = 0;
capelin_called_red = 0;
capelin_called_cod = 0;
capelin_called_cap = 0;
redfish_called_red = 0;
redfish_called_cod = 0;
redfish_called_cap = 0;

num_cod = length(norm_cod);
num_capelin = length(norm_capelin);
num_redfish = length(norm_red);


% list of features selected
% Node 1 - extract redfish
keep1 = [2 3 4 6 7 11 12 15 16];
```

```
% Node 2 - cod vs. capelin
keep2 =    [3 4 6 7 9 12 15 20];


for i = 1:length(alldata)

   M_cod1 = mean(norm_cod(:,keep1));
   M_capelin1 = mean(norm_capelin(:,keep1));
   M_red1 = mean(norm_red(:,keep1));

   K_cod1 = cov(norm_cod(:,keep1));
   K_capelin1 = cov(norm_capelin(:,keep1));
   K_red1 = cov(norm_red(:,keep1));

   % leave out the sample we are testing
   if( i<=num_cod)
      M_cod1 = mean(norm_cod(find(1:num_cod~=i),keep1));
      K_cod1 = cov(norm_cod(find(1:num_cod~=i),keep1));
   elseif (i > num_cod) & (i <= (num_cod + num_capelin))
      M_capelin1 = mean(norm_capelin(find(1:num_capelin~=(i-num_cod)),keep1));
      K_capelin1 = cov(norm_capelin(find(1:num_capelin~=(i-num_cod)),keep1));
   else
      M_red1 = mean(norm_red(find(1:num_redfish~=(i-num_cod-num_capelin)),keep1));
      K_red1 = cov(norm_red(find(1:num_redfish~=(i-num_cod-num_capelin)),keep1));
   end

    data = [alldata(i,keep1)];
    dist_cod = (data - M_cod1) * inv(K_cod1) * (data - M_cod1)';
    dist_capelin = (data - M_capelin1) * inv(K_capelin1) * (data - M_capelin1)';
    dist_red = (data - M_red1) * inv(K_red1) * (data - M_red1)';

   if ( (dist_red < dist_cod) & (dist_red< dist_capelin))
        called = 3; % 3 = redfish;
   else
        data = [alldata(i,keep2)];
            sameCount = 0;
        for j = 1:length(node2data)
                  tmp = data - node2data(j,keep2);
            dist2(j) = sqrt(tmp*tmp');
            if (dist2(j) == 0) % the same point so don't use
               dist2(j) == 25;
               sameCount = sameCount + 1;
               if sameCount > 1
                  fprintf('Error - too many identical points\n ');
                  pause;
               end
           end
        end
        [sortedDist, Index] = sort(dist2);

        if ( length(find(Index(1:3) <= num_cod )) >= 2) % 2/3 were cod
                called = 1; % 1 = cod
            else
                called = 2; % 2 = capelin
        end

      end  % end of Node 2
```

```
      if called == 1
            if (i <= num_cod)
                  cod_called_cod = cod_called_cod + 1;
            elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                  capelin_called_cod = capelin_called_cod + 1;
            else
                  redfish_called_cod = redfish_called_cod + 1;
      end

      elseif called == 2
            if (i <= num_cod)
                  cod_called_cap = cod_called_cap + 1;
            elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                  capelin_called_cap = capelin_called_cap + 1;
            else
                  redfish_called_cap = redfish_called_cap + 1;
            end

      else
            if (i <= num_cod)
                  cod_called_red = cod_called_red + 1;
            elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                  capelin_called_red = capelin_called_red + 1;
            else
                  redfish_called_red = redfish_called_red + 1;
            end
   end

end

fprintf('Node #1 Number of Features = %d\nFeature List = ',length(keep1));
fprintf('%d  ', keep1);

fprintf('\nNode #2 Number of Features = %d\nFeature List = ',length(keep2));
fprintf('%d  ', keep2);

fprintf('\n\t\t\t\tPREDICTED CLASS\n');
fprintf('\t\tCOD\t\tCAPELIN\t\tREDFISH\n');

fprintf('COD\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n',...
cod_called_cod,cod_called_cod/num_cod*100,...
cod_called_cap,cod_called_cap/num_cod*100,...
cod_called_red,cod_called_red/num_cod*100);

fprintf('CAPELIN\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n',...
capelin_called_cod,capelin_called_cod/num_capelin*100,...
capelin_called_cap,capelin_called_cap/num_capelin*100,...
capelin_called_red,capelin_called_red/num_capelin*100);

fprintf('REDFISH\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n\n',...
redfish_called_cod,redfish_called_cod/num_redfish*100,...
redfish_called_cap,redfish_called_cap/num_redfish*100,...
redfish_called_red,redfish_called_red/num_redfish*100);
```

```
% sbs3nn.m
% sequential backward selection for the
% 3-nearest neighbour classifiers for cod, capelin, and redfish
%
% Patricia LeFeuvre

clear,
load capelin.txt;
load cod.txt;
load redfish.txt;

%normalize all of the data
MN_red = min(redfish);
MN_capelin = min(capelin);
MN_cod = min(cod);
MN = min(min(MN_red,MN_capelin),MN_cod);

MX_red = max(redfish);
MX_capelin = max(capelin);
MX_cod = max(cod);
MX = max(max(MX_red,MX_capelin),MX_cod)-MN;

for i = 1:length(redfish)
      norm_red(i,1:25) = (redfish(i,1:25) - MN)./MX;
end

for i = 1:length(capelin)
      norm_capelin(i,1:25) = (capelin(i,1:25) - MN)./MX;
end

for i = 1:length(cod)
      norm_cod(i,1:25) = (cod(i,1:25) - MN)./MX;
end

% Classifier #1 or stage 1 of the other three classifiers
alldata = [norm_cod; norm_capelin; norm_red];

% Classifier #4 - stage 2 cod vs capelin
%alldata = [norm_cod; norm_capelin];


num_cod = length(norm_cod);
num_capelin = length(norm_capelin);
num_redfish = length(norm_red);


%% CLASSIFIER
%keep1 = [1 3 4 5 6 7 9 10 11 12 15 16 17 18 19 20 22 24 25];     % After depth
feature removal
%keepList = [1 0 1 1 1 1 1 0 1 1 1 1 0 0 1 1 1 1 1 1 0 1 0 1 1];  % from Factor
Analysis results

% list of features still in the running
keep1 =    [1 2 3 4 5 6 7    9  10 11 12       15 16          20    22    24 25];
      % After depth feature removal and fact Ana
keepList = [1 1 1 1 1 1 1 0  1  1  1  1  0  0  1  1  0  0  0  1  0  1  0  1  1];
      % no feret features
```

```
%keep track of best performance for each number of features
for i = 1:17
      bestForNumFeatures(i) = 0;
end

for num_removed = 0:16

      performance = zeros(25,1);
      for feat = 1:25
            if(keepList(feat) == 1)
                  newkeepList = keepList;

            if (num_removed > 0)
                        newkeepList(feat) = 0;
            end
                  keep1 = find(newkeepList == 1);
                  cod_called_red = 0;
                  cod_called_cod = 0;
                  cod_called_cap = 0;
                  capelin_called_red = 0;
                  capelin_called_cod = 0;
                  capelin_called_cap = 0;
                  redfish_called_red = 0;
                  redfish_called_cod = 0;
                  redfish_called_cap = 0;

                  for i = 1:length(alldata)
                        data = [alldata(i,keep1)];

                        for j = 1:length(alldata)
                              if i ~= j
                                    tmp = data - alldata(j,keep1);
                                    dist(j) = sqrt(tmp*tmp');
                              else
                                    dist(j) = 25;
                              end
                        end

                        [sortedDist, Index] = sort(dist);

                        if ( length(find(Index(1:3) <= num_cod )) >= 2) % 2/3 were
cod
                              called = 1; % 1 = cod;
                        elseif ( length(find( (Index(1:3) > num_cod) & (Index(1:3)
<= (num_cod+num_capelin))) ) >= 2) % 2/3 were capelin
                              called = 2; % 2 = capelin
                  elseif ( length(find(Index(1:3) > (num_cod+num_capelin))) >= 2) %
2/3 were redfish
                              called = 3; % 3 = redfish
                        else
                              %3 way tie - use the distance to the nearest
neighbour
                              if (Index(1) <= num_cod)
                                    called = 1;
                              elseif (Index(1) <= (num_cod+num_capelin))
                                    called = 2;
                              else
                                    called = 3;
```

```
                        end
                    end
                    if called == 1
                        if (i <= num_cod)
                            cod_called_cod = cod_called_cod + 1;
                        elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                            capelin_called_cod = capelin_called_cod + 1;
                        else
                            redfish_called_cod = redfish_called_cod + 1;
                        end
                    elseif called == 2
                        if (i <= num_cod)
                            cod_called_cap = cod_called_cap + 1;
                        elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                            capelin_called_cap = capelin_called_cap + 1;
                        else
                            redfish_called_cap = redfish_called_cap + 1;
                        end
                    else
                        if (i <= num_cod)
                            cod_called_red = cod_called_red + 1;
                        elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                            capelin_called_red = capelin_called_red + 1;
                        else
                            redfish_called_red = redfish_called_red + 1;
                        end
                    end
                end

            %Classifier #1
            %performance(feat) = (cod_called_cod/num_cod) +
(capelin_called_cap/num_capelin) + (redfish_called_red/num_redfish);

            %Classifier #2 - stage 1 cod detection
            %performance(feat) = (cod_called_cod/num_cod) +
(1-capelin_called_cod/num_capelin) + (1-redfish_called_cod/num_redfish);
            %Classifier #2 - stage 2 capelin vs redfish detection
          %performance(feat) = capelin_called_cap/num_capelin +
redfish_called_red/num_redfish;

            %Classifier #3 - stage 1 capelin detection
            %performance(feat) = (1-cod_called_cap/num_cod) +
(capelin_called_cap/num_capelin) + (1-redfish_called_cap/num_redfish);
            %Classifier #3 - stage 2 cod vs redfish detection
            %performance(feat) = cod_called_cod/num_cod  +
redfish_called_red/num_redfish;

            %Classifier #4 - stage 1 redfish detection
            %performance(feat) = (1-cod_called_red/num_cod) +
(1-capelin_called_red/num_capelin) + (redfish_called_red/num_redfish);
            %Classifier #4 - stage 2 cod vs capelin detection
            performance(feat) = cod_called_cod/num_cod  +
capelin_called_cap/num_capelin;

            end   % if(keepList(feat) == 1)
        end   % for feat = 1:25

        [BestPerformance, I] = max(performance);
```

```
    ind = length(keep1);
      bestForNumFeatures(ind) = BestPerformance;

    if (num_removed > 0)
      keepList(I) = 0;
    end

      fprintf('\nnum Features removed = %d, performance = %.3f\n', num_removed,
BestPerformance);

      keep1 = find(keepList == 1);

      cod_called_red = 0;
      cod_called_cod = 0;
      cod_called_cap = 0;
      capelin_called_red = 0;
      capelin_called_cod = 0;
      capelin_called_cap = 0;
      redfish_called_red = 0;
      redfish_called_cod = 0;
      redfish_called_cap = 0;

      for i = 1:length(alldata)
            data = [alldata(i,keep1)];

            for j = 1:length(alldata)
                  if i ~= j
                        tmp = data - alldata(j,keep1);
                        dist(j) = sqrt(tmp*tmp');
                  else
                        dist(j) = 25;
                  end
            end

            [sortedDist, Index] = sort(dist);
            if ( length(find(Index(1:3) <= num_cod )) >= 2) % 2/3 were cod
                  called = 1; % 1 = cod;
            elseif ( length(find( (Index(1:3) > num_cod) & (Index(1:3) <=
(num_cod+num_capelin))) ) >= 2) % 2/3 were capelin
                  called = 2; % 2 = capelin
            elseif ( length(find(Index(1:3) > (num_cod+num_capelin))) >= 2) % 2/3
were redfish
                  called = 3; % 3 = capelin
            else
                  %3 way tie - use the distance to the nearest neighbour
                  if (Index(1) <= num_cod)
                        called = 1;
                  elseif (Index(1) <= (num_cod+num_capelin))
                        called = 2;
                  else
                        called = 3;
                  end
            end

            if called == 1
                  if (i <= num_cod)
                        cod_called_cod = cod_called_cod + 1;
```

```
                elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                        capelin_called_cod = capelin_called_cod + 1;
                else
                        redfish_called_cod = redfish_called_cod + 1;
                end

        elseif called == 2
                if (i <= num_cod)
                        cod_called_cap = cod_called_cap + 1;
                elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                        capelin_called_cap = capelin_called_cap + 1;
                else
                        redfish_called_cap = redfish_called_cap + 1;
                end

        else
                if (i <= num_cod)
                        cod_called_red = cod_called_red + 1;
                elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                        capelin_called_red = capelin_called_red + 1;
                else
                        redfish_called_red = redfish_called_red + 1;
                end
        end
    end

    fprintf('Number of Features = %d\nFeature List = ',length(keep1));
    fprintf('%d  ', keep1);
    fprintf('\n\t\t\t\tPREDICTED CLASS\n');
    fprintf('\t\tCOD\t\tCAPELIN\t\tREDFISH\n');
    fprintf('COD\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n',...
    cod_called_cod,cod_called_cod/num_cod*100,...
    cod_called_cap,cod_called_cap/num_cod*100,...
    cod_called_red,cod_called_red/num_cod*100);
    fprintf('CAPELIN\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n',...
    capelin_called_cod,capelin_called_cod/num_capelin*100,...
    capelin_called_cap,capelin_called_cap/num_capelin*100,...
    capelin_called_red,capelin_called_red/num_capelin*100);
    fprintf('REDFISH\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n\n',...
    redfish_called_cod,redfish_called_cod/num_redfish*100,...
    redfish_called_cap,redfish_called_cap/num_redfish*100,...
    redfish_called_red,redfish_called_red/num_redfish*100);
    pause(1);


end   % num_removed = 1:18


plot(bestForNumFeatures)
xlabel('number of features used')
grid
ylabel('average classification accuracy (percentage)')
```

```
% sfs3nn.m
% sequential forward selection for the
% 3-nearest neighbour classifiers for cod, capelin, and redfish
%
% Patricia LeFeuvre

clear,
load capelin.txt;
load cod.txt;
load redfish.txt;

%normalize all of the data
MN_red = min(redfish);
MN_capelin = min(capelin);
MN_cod = min(cod);
MN = min(min(MN_red,MN_capelin),MN_cod);

MX_red = max(redfish);
MX_capelin = max(capelin);
MX_cod = max(cod);
MX = max(max(MX_red,MX_capelin),MX_cod)-MN;

for i = 1:length(redfish)
      norm_red(i,1:25) = (redfish(i,1:25) - MN)./MX;
end

for i = 1:length(capelin)
      norm_capelin(i,1:25) = (capelin(i,1:25) - MN)./MX;
end

for i = 1:length(cod)
      norm_cod(i,1:25) = (cod(i,1:25) - MN)./MX;
end

% Classifier #1 or stage 1 of the other three classifiers
alldata = [norm_cod; norm_capelin; norm_red];

% Classifier #4 - stage 2 cod vs capelin
%alldata = [norm_cod; norm_capelin];


num_cod = length(norm_cod);
num_capelin = length(norm_capelin);
num_redfish = length(norm_red);


% list of features still in the running
%keep1 = [1 3 4 5 6 7 9 10 11 12 15 16 17 18 19 20 22 24 25];     % After depth
feature removal
%keepList1 = [1 0 1 1 1 1 1 0 1 1 1 1 1 0 0 1 1 1 1 1 1 1 0 1 0 1 1]; % from Factor
Analysis results
% list of features still in the running
keep1 =     [1 2 3 4 5 6 7    9  10 11 12       15 16          20    22    24 25];
      % After depth feature removal and fact Ana
keepList1 = [1 1 1 1 1 1 1 0  1  1  1  1  0  0  1  1  0  0  0  1  0  1  0  1  1];
      % no feret features

% starting point
```

```
keep = [];
keepList =  [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];

%keep track of best performance for each number of features
for i = 1:18
      bestForNumFeatures(i) = 0;
end

for num_added = 1:11

      performance = zeros(25,1);

      for feat = 1:25
      if ((keepList(feat) == 0) & (keepList1(feat) == 1))

                  newKeepList = keepList;
                  newKeepList(feat) = 1;
                  keep1 = find(newKeepList == 1);

                  cod_called_red = 0;
                  cod_called_cod = 0;
                  cod_called_cap = 0;
                  capelin_called_red = 0;
                  capelin_called_cod = 0;
                  capelin_called_cap = 0;
                  redfish_called_red = 0;
                  redfish_called_cod = 0;
                  redfish_called_cap = 0;

                  for i = 1:length(alldata)
                        data = [alldata(i,keep1)];

                        for j = 1:length(alldata)
                              if i ~= j
                                    tmp = data - alldata(j,keep1);
                                    dist(j) = sqrt(tmp*tmp');
                              else
                                    dist(j) = 25;
                              end
                        end

                        [sortedDist, Index] = sort(dist);

                        if ( length(find(Index(1:3) <= num_cod )) >= 2) % 2/3 were
cod
                              called = 1; % 1 = cod;
                        elseif ( length(find( (Index(1:3) > num_cod) & (Index(1:3)
<= (num_cod+num_capelin))) ) >= 2) % 2/3 were capelin
                              called = 2; % 2 = capelin
                        %elseif ( length(find(Index(1:3) > (num_cod+num_capelin)))
>= 2) % 2/3 were redfish
                        %     called = 3; % 3 = redfish
                        else
                              %3 way tie - use the distance to the nearest
neighbour
                              if (Index(1) <= num_cod)
                                    called = 1;
                              elseif (Index(1) <= (num_cod+num_capelin))
```

```
                        called = 2;
                    else
                        called = 3;
                    end
                end

                if called == 1
                    if (i <= num_cod)
                        cod_called_cod = cod_called_cod + 1;
                    elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                        capelin_called_cod = capelin_called_cod + 1;
                    else
                        redfish_called_cod = redfish_called_cod + 1;
                    end

                elseif called == 2
                    if (i <= num_cod)
                        cod_called_cap = cod_called_cap + 1;
                    elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                        capelin_called_cap = capelin_called_cap + 1;
                    else
                        redfish_called_cap = redfish_called_cap + 1;
                    end

                else
                    if (i <= num_cod)
                        cod_called_red = cod_called_red + 1;
                    elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                        capelin_called_red = capelin_called_red + 1;
                    else
                        redfish_called_red = redfish_called_red + 1;
                    end

                end
            end

        %Classifier #1
        %performance(feat) = (cod_called_cod/num_cod) +
(capelin_called_cap/num_capelin) + (redfish_called_red/num_redfish);

        %Classifier #2 - stage 1 cod detection
        %performance(feat) = (cod_called_cod/num_cod) +
(1-capelin_called_cod/num_capelin) + (1-redfish_called_cod/num_redfish);
        %Classifier #2 - stage 2 capelin vs redfish detection
        %performance(feat) = capelin_called_cap/num_capelin +
redfish_called_red/num_redfish;

        %Classifier #3 - stage 1 capelin detection
        %performance(feat) = (1-cod_called_cap/num_cod) +
(capelin_called_cap/num_capelin) + (1-redfish_called_cap/num_redfish);
        %Classifier #3 - stage 2 cod vs redfish detection
        %performance(feat) = cod_called_cod/num_cod  +
redfish_called_red/num_redfish;

        %Classifier #4 - stage 1 redfish detection
        %performance(feat) = (1-cod_called_red/num_cod) +
(1-capelin_called_red/num_capelin) + (redfish_called_red/num_redfish);
        %Classifier #4 - stage 2 cod vs capelin detection
```

```
            performance(feat) = cod_called_cod/num_cod  +
capelin_called_cap/num_capelin;


            end   % if(keepList(feat) == 1)
      end   % for feat = 1:25

      [BestPerformance, I] = max(performance);
      bestForNumFeatures(num_added) = BestPerformance;
   keepList(I) = 1;
      keep1 = find(keepList == 1);

   fprintf('\nnum Features used = %d, performance = %.3f\n', num_added,
BestPerformance);

      cod_called_red = 0;
      cod_called_cod = 0;
      cod_called_cap = 0;
      capelin_called_red = 0;
      capelin_called_cod = 0;
      capelin_called_cap = 0;
      redfish_called_red = 0;
      redfish_called_cod = 0;
      redfish_called_cap = 0;

      for i = 1:length(alldata)
            data = [alldata(i,keep1)];

            for j = 1:length(alldata)
                  if i ~= j
                        tmp = data - alldata(j,keep1);
                        dist(j) = sqrt(tmp*tmp');
                  else
                        dist(j) = 25;
                  end
            end

            [sortedDist, Index] = sort(dist);
            if ( length(find(Index(1:3) <= num_cod )) >= 2) % 2/3 were cod
                  called = 1; % 1 = cod;
            elseif ( length(find( (Index(1:3) > num_cod) & (Index(1:3) <=
(num_cod+num_capelin))) ) >= 2) % 2/3 were capelin
                  called = 2; % 2 = capelin
            elseif ( length(find(Index(1:3) > (num_cod+num_capelin))) >= 2) % 2/3
were redfish
                  called = 3; % 3 = capelin
            else
                  %3 way tie - use the distance to the nearest neighbour
                  if (Index(1) <= num_cod)
                        called = 1;
                  elseif (Index(1) <= (num_cod+num_capelin))
                        called = 2;
                  else
                        called = 3;
                  end
            end

            if called == 1
```

```
                if (i <= num_cod)
                        cod_called_cod = cod_called_cod + 1;
                elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                        capelin_called_cod = capelin_called_cod + 1;
                else
                        redfish_called_cod = redfish_called_cod + 1;
                end

        elseif called == 2
                if (i <= num_cod)
                        cod_called_cap = cod_called_cap + 1;
                elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                        capelin_called_cap = capelin_called_cap + 1;
                else
                        redfish_called_cap = redfish_called_cap + 1;
                end

        else
                if (i <= num_cod)
                        cod_called_red = cod_called_red + 1;
                elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                        capelin_called_red = capelin_called_red + 1;
                else
                        redfish_called_red = redfish_called_red + 1;
                end
        end
    end

    fprintf('Number of Features = %d\nFeature List = ',length(keep1));
    fprintf('%d  ', keep1);
    fprintf('\n\t\t\t\tPREDICTED CLASS\n');
    fprintf('\t\tCOD\t\tCAPELIN\t\tREDFISH\n');

    fprintf('COD\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n',...
    cod_called_cod,cod_called_cod/num_cod*100,...
    cod_called_cap,cod_called_cap/num_cod*100,...
    cod_called_red,cod_called_red/num_cod*100);

    fprintf('CAPELIN\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n',...
    capelin_called_cod,capelin_called_cod/num_capelin*100,...
    capelin_called_cap,capelin_called_cap/num_capelin*100,...
    capelin_called_red,capelin_called_red/num_capelin*100);

    fprintf('REDFISH\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n\n',...
    redfish_called_cod,redfish_called_cod/num_redfish*100,...
    redfish_called_cap,redfish_called_cap/num_redfish*100,...
    redfish_called_red,redfish_called_red/num_redfish*100);

    pause(1);

end   % num_removed = 1:18
```

```
% SBSMal.m
%
%      Patricia LeFeuvre
%      Feb 19, 2001
%
%      Program to perform sequental backward selection to find the best
%      10 or fewer features to distinguish capelin from cod and redfish
%      Using Thesis data

clear,
load capelin.txt;
load cod.txt;
load redfish.txt;

% Normalize
MN_red = min(redfish);
MN_capelin = min(capelin);
MN_cod = min(cod);
MN = min(min(MN_red,MN_capelin),MN_cod);

MX_red = max(redfish);
MX_capelin = max(capelin);
MX_cod = max(cod);
MX = max(max(MX_red,MX_capelin),MX_cod)-MN;

for i = 1:length(redfish)
      norm_red(i,1:25) = (redfish(i,1:25) - MN)./MX;
end

for i = 1:length(capelin)
      norm_capelin(i,1:25) = (capelin(i,1:25) - MN)./MX;
end

for i = 1:length(cod)
      norm_cod(i,1:25) = (cod(i,1:25) - MN)./MX;
end

alldata = [norm_cod; norm_capelin; norm_red];


num_cod = length(norm_cod);
num_capelin = length(norm_capelin);
num_redfish = length(norm_red);

% list of features still in the running
%keep1 = [1 3 4 5 6 7 9 10 11 12 15 16 17 18 19 20 22 24 25];     % After depth
feature removal
%keepList = [1 0 1 1 1 1 1 0 1 1 1 1 0 0 1 1 1 1 1 1 0 1 0 1 1]; % from Factor
Analysis results

% list of features still in the running
keep1 =     [1 2 3 4 5 6 7    9  10 11 12       15 16          20    22    24 25];
      % After depth feature removal and fact Ana
keepList = [1 1 1 1 1 1 1 0  1  1  1  1  0  0  1  1  0  0  0  1  0  1  0  1  1];
      % no feret features

training_cod = norm_cod;                    %[norm_cod(:,keep1)];
training_capelin = norm_capelin; %[norm_capelin(:,keep1)];
```

```
training_red = norm_red;                    %[norm_red(:,keep1)];

%  to illustrate over training
for i = 1:17
      bestForNumFeatures(i) = 0;
end

for num_removed = 0:16  %0:16

      performance = zeros(25,1);

      for feat = 1:25

      if (keepList(feat) == 1)

                  newKeepList = keepList;
            if num_removed > 0
                        newKeepList(feat) = 0;
            end
                  keep1 = find(newKeepList == 1);

            cod_called_red = 0;
                  cod_called_cod = 0;
                  cod_called_cap = 0;
                  capelin_called_red = 0;
                  capelin_called_cod = 0;
                  capelin_called_cap = 0;
                  redfish_called_red = 0;
                  redfish_called_cod = 0;
                  redfish_called_cap = 0;

                  M_cod = mean(training_cod(:,keep1));
                  M_capelin = mean(training_capelin(:,keep1));
            M_red = mean(training_red(:,keep1));

                  K_cod = cov(training_cod(:,keep1));
                  K_capelin = cov(training_capelin(:,keep1));
                  K_red = cov(training_red(:,keep1));

                  for i = 1:length(alldata)
                        data = [alldata(i,keep1)];
            dist_cod = (data - M_cod) * inv(K_cod) * (data - M_cod)';
            dist_capelin = (data - M_capelin) * inv(K_capelin) * (data -
M_capelin)';
            dist_red = (data - M_red) * inv(K_red) * (data - M_red)';
            if ( (dist_cod < dist_capelin) & (dist_cod < dist_red)  )
               called = 1; % 1 = cod;
            elseif ( (dist_capelin < dist_cod) & (dist_capelin < dist_red))
               called = 2; % 2 = capelin
                        elseif ((dist_red < dist_cod) & (dist_red < dist_capelin) )
               called = 3; % 3 = redfish
            else
               called = 4; % we have a tie
               fprintf('\nwe have a tie with the distances\n');
               pause;
            end

            if called == 1
```

```
                        if (i <= num_cod)
                              cod_called_cod = cod_called_cod + 1;
                        elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                              capelin_called_cod = capelin_called_cod + 1;
                        else
                              redfish_called_cod = redfish_called_cod + 1;
                        end
                  elseif called == 2
                        if (i <= num_cod)
                              cod_called_cap = cod_called_cap + 1;
                        elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                              capelin_called_cap = capelin_called_cap + 1;
                        else
                              redfish_called_cap = redfish_called_cap + 1;
                        end
                  elseif called == 3
                        if (i <= num_cod)
                              cod_called_red = cod_called_red + 1;
                        elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                              capelin_called_red = capelin_called_red + 1;
                        else
                              redfish_called_red = redfish_called_red + 1;
            end
         end

               end % end of i = 1:length(alldata)

      %Classifier #5
               %performance(feat) = (cod_called_cod/num_cod) +
(capelin_called_cap/num_capelin) + (redfish_called_red/num_redfish);

      %Classifier #6 - stage 1 cod detection
               %performance(feat) = (cod_called_cod/num_cod) +
(1-capelin_called_cod/num_capelin) + (1-redfish_called_cod/num_redfish);
               %Classifier #6 - stage 2 capelin vs redfish detection
               %performance(feat) = capelin_called_cap/num_capelin +
redfish_called_red/num_redfish;

               %Classifier #7 - stage 1 capelin detection
               %performance(feat) = (1-cod_called_cap/num_cod) +
(capelin_called_cap/num_capelin) + (1-redfish_called_cap/num_redfish);
               %Classifier #7 - stage 2 cod vs redfish detection
               %performance(feat) = cod_called_cod/num_cod  +
redfish_called_red/num_redfish;

               %Classifier #8 - stage 1 redfish detection
               %performance(feat) = (1-cod_called_red/num_cod) +
(1-capelin_called_red/num_capelin) + (redfish_called_red/num_redfish);
               %Classifier #8 - stage 2 cod vs capelin detection
               performance(feat) = cod_called_cod/num_cod  +
capelin_called_cap/num_capelin;

   end % end of if (keepList(feat) == 1)

   end   % end of for feat = 1:25


   %performance,
```

```
      [BestPerformance,Indx] = max(performance);
    fprintf('\nnum Features removed = %d, performance = %.3f\n', num_removed,
BestPerformance);

    % to illustrate over training
    ind = length(keep1);
    perf_per_num_feats(ind) = BestPerformance/3.00*100;

    if num_removed > 0
        keepList(Indx) = 0;
    end
      keep1 = find(keepList == 1);

      cod_called_red = 0;
      cod_called_cod = 0;
      cod_called_cap = 0;
      capelin_called_red = 0;
      capelin_called_cod = 0;
      capelin_called_cap = 0;
      redfish_called_red = 0;
      redfish_called_cod = 0;
    redfish_called_cap = 0;

      M_cod = mean(training_cod(:,keep1));
      M_capelin = mean(training_capelin(:,keep1));
    M_red = mean(training_red(:,keep1));

      K_cod = cov(training_cod(:,keep1));
      K_capelin = cov(training_capelin(:,keep1));
      K_red = cov(training_red(:,keep1));

      for i = 1:length(alldata)
            data = [alldata(i,keep1)];

            dist_cod = (data - M_cod) * inv(K_cod) * (data - M_cod)';
            dist_capelin = (data - M_capelin) * inv(K_capelin) * (data -
M_capelin)';
            dist_red = (data - M_red) * inv(K_red) * (data - M_red)';
            if ( (dist_cod < dist_capelin) & (dist_cod < dist_red)  )
                  called = 1; % 1 = cod;
            elseif ( (dist_capelin < dist_cod) & (dist_capelin < dist_red))
            called = 2; % 2 = capelin
            elseif ((dist_red < dist_cod) & (dist_red < dist_capelin) )
            called = 3; % 3 = redfish
            else
            called = 4; % we have a tie
            fprintf('\nwe have a tie with the distances\n');
            pause;
            end

            if called == 1
            if (i <= num_cod)
             cod_called_cod = cod_called_cod + 1;
            elseif (i > num_cod) & (i <= (num_cod + num_capelin))
             capelin_called_cod = capelin_called_cod + 1;
                  else
              redfish_called_cod = redfish_called_cod + 1;
             end
```

```
            elseif called == 2
            if (i <= num_cod)
                  cod_called_cap = cod_called_cap + 1;
            elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                  capelin_called_cap = capelin_called_cap + 1;
            else
                  redfish_called_cap = redfish_called_cap + 1;
            end

            elseif called == 3
                  if (i <= num_cod)
                  cod_called_red = cod_called_red + 1;
            elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                  capelin_called_red = capelin_called_red + 1;
            else
                  redfish_called_red = redfish_called_red + 1;
            end
            end
      end

      fprintf('Number of Features = %d\nFeature List = ',length(keep1));
      fprintf('%d  ', keep1);
      fprintf('\n\t\t\t\tPREDICTED CLASS\n');
      fprintf('\t\tCOD\t\tCAPELIN\t\tREDFISH\n');

      fprintf('COD\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n',...
      cod_called_cod,cod_called_cod/num_cod*100,...
      cod_called_cap,cod_called_cap/num_cod*100,...
      cod_called_red,cod_called_red/num_cod*100);

      fprintf('CAPELIN\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n',...
      capelin_called_cod,capelin_called_cod/num_capelin*100,...
      capelin_called_cap,capelin_called_cap/num_capelin*100,...
      capelin_called_red,capelin_called_red/num_capelin*100);

      fprintf('REDFISH\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n\n',...
      redfish_called_cod,redfish_called_cod/num_redfish*100,...
      redfish_called_cap,redfish_called_cap/num_redfish*100,...
      redfish_called_red,redfish_called_red/num_redfish*100);

%     perf = (cod_called_cod/num_cod) + (capelin_called_cap/num_capelin) +
(redfish_called_red/num_redfish),

end

plot(perf_per_num_feats)
xlabel('number of features used')
grid
ylabel('average classification accuracy (percentage)')
```

```
% SFSMal.m
%
%      Patricia LeFeuvre
%      Feb 26, 2001
%
%      Program to perform sequental forward selection to find the best
%      10 or fewer features to distinguish capelin from cod and redfish
%      Using Thesis data

clear,
load capelin.txt;
load cod.txt;
load redfish.txt;

% Normalize
MN_red = min(redfish);
MN_capelin = min(capelin);
MN_cod = min(cod);
MN = min(min(MN_red,MN_capelin),MN_cod);

MX_red = max(redfish);
MX_capelin = max(capelin);
MX_cod = max(cod);
MX = max(max(MX_red,MX_capelin),MX_cod)-MN;

for i = 1:length(redfish)
      norm_red(i,1:25) = (redfish(i,1:25) - MN)./MX;
end

for i = 1:length(capelin)
      norm_capelin(i,1:25) = (capelin(i,1:25) - MN)./MX;
end

for i = 1:length(cod)
      norm_cod(i,1:25) = (cod(i,1:25) - MN)./MX;
end

alldata = [norm_cod; norm_capelin; norm_red];


num_cod = length(norm_cod);
num_capelin = length(norm_capelin);
num_redfish = length(norm_red);

% list of features still in the running
%keep1 =     [1   3 4 5 6 7    9  10 11 12       15 16 17 18 19 20    22   24 25];
      % After depth feature removal
%keepList1 = [1 0 1 1 1 1 1 0  1  1  1  0  0  1  1  1  1  1  1  0  1  0  1  1];
      % from Factor Analysis results

keep1 =     [1 2 3 4 5 6 7    9  10 11 12       15 16          20    22    24 25];
      % After depth feature removal and fact Ana
keepList1 = [1 1 1 1 1 1 1 0  1  1  1  1  0  0  1  1  0  0  0  1  0  1  0  1  1];
      % no feret features


% temp to illustrate overtraining
%keep1 =     [1 2 3 4 5 6 7 8 9 10 11 12      15 16 17 18 19 20 21 22 23 24 25];
```

```
      % After depth feature removal
%keepList1 = [1 1 1 1 1 1 1 1 1 1 1  1  1  0  0 1  1  1  1  1  1  1  1  1  1  1];
      % from Factor Analysis results

% starting point
keep = [];
keepList =  [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];

%% CLASSIFIER #5 - Distinguish cod from capelin from Redfish
training_cod = norm_cod;                    %[norm_cod(:,keep1)];
training_capelin = norm_capelin;     %[norm_capelin(:,keep1)];
training_red = norm_red;                    %[norm_red(:,keep1)];

% temp to illustrate overtraining
for i = 1:17
      bestForNumFeatures(i) = 0;
end

for num_added = 1:17

      performance = zeros(25,1);

      for feat = 1:25

      if ((keepList(feat) == 0) & (keepList1(feat) == 1))

                  newKeepList = keepList;
                  newKeepList(feat) = 1;
                  keep1 = find(newKeepList == 1);

            cod_called_red = 0;
                  cod_called_cod = 0;
                  cod_called_cap = 0;
                  capelin_called_red = 0;
                  capelin_called_cod = 0;
                  capelin_called_cap = 0;
                  redfish_called_red = 0;
                  redfish_called_cod = 0;
                  redfish_called_cap = 0;

                  M_cod = mean(training_cod(:,keep1));
                  M_capelin = mean(training_capelin(:,keep1));
            M_red = mean(training_red(:,keep1));

                  K_cod = cov(training_cod(:,keep1));
                  K_capelin = cov(training_capelin(:,keep1));
                  K_red = cov(training_red(:,keep1));


                  for i = 1:length(alldata)
                        data = [alldata(i,keep1)];
            dist_cod = (data - M_cod) * inv(K_cod) * (data - M_cod)';
            dist_capelin = (data - M_capelin) * inv(K_capelin) * (data -
M_capelin)';
            dist_red = (data - M_red) * inv(K_red) * (data - M_red)';
            if ( (dist_cod < dist_capelin) & (dist_cod < dist_red)  )
               called = 1; % 1 = cod;
            elseif ( (dist_capelin < dist_cod) & (dist_capelin < dist_red))
```

```
            called = 2; % 2 = capelin
                    elseif ((dist_red < dist_cod) & (dist_red < dist_capelin) )
            called = 3; % 3 = redfish
        else
            called = 4; % we have a tie
            printf('\nwe have a tie with the distances\n');
            pause;
        end

        if called == 1
                        if (i <= num_cod)
                            cod_called_cod = cod_called_cod + 1;
                        elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                            capelin_called_cod = capelin_called_cod + 1;
                        else
                            redfish_called_cod = redfish_called_cod + 1;
                        end
                    elseif called == 2
                        if (i <= num_cod)
                            cod_called_cap = cod_called_cap + 1;
                        elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                            capelin_called_cap = capelin_called_cap + 1;
                        else
                            redfish_called_cap = redfish_called_cap + 1;
                        end
                    elseif called == 3
                        if (i <= num_cod)
                            cod_called_red = cod_called_red + 1;
                        elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                            capelin_called_red = capelin_called_red + 1;
                        else
                            redfish_called_red = redfish_called_red + 1;
            end
        end

              end % end of i = 1:length(alldata)

    %Classifier #5
              %performance(feat) = (cod_called_cod/num_cod) +
(capelin_called_cap/num_capelin) + (redfish_called_red/num_redfish);

    %Classifier #6 - stage 1 cod detection
              %performance(feat) = (cod_called_cod/num_cod) +
(1-capelin_called_cod/num_capelin) + (1-redfish_called_cod/num_redfish);
              %Classifier #6 - stage 2 capelin vs redfish detection
              %performance(feat) = capelin_called_cap/num_capelin +
redfish_called_red/num_redfish;

              %Classifier #7 - stage 1 capelin detection
              %performance(feat) = (1-cod_called_cap/num_cod) +
(capelin_called_cap/num_capelin) + (1-redfish_called_cap/num_redfish);
              %Classifier #7 - stage 2 cod vs redfish detection
              %performance(feat) = cod_called_cod/num_cod  +
redfish_called_red/num_redfish;

              %Classifier #8 - stage 1 redfish detection
              %performance(feat) = (1-cod_called_red/num_cod) +
(1-capelin_called_red/num_capelin) + (redfish_called_red/num_redfish);
```

```
                %Classifier #8 - stage 2 cod vs capelin detection
                performance(feat) = cod_called_cod/num_cod  +
capelin_called_cap/num_capelin;

      end % end of if (keepList(feat) == 1)

      end   % end of for feat = 1:25


      %performance,
      [BestPerformance,Indx] = max(performance);
   fprintf('\nnum Features used = %d, performance = %.3f\n', num_added,
BestPerformance);

   % temp to illustrate overtraining
   perf_per_num_feats(num_added) = BestPerformance/3.00*100;


   keepList(Indx) = 1;
      keep1 = find(keepList == 1);

      cod_called_red = 0;
      cod_called_cod = 0;
      cod_called_cap = 0;
      capelin_called_red = 0;
      capelin_called_cod = 0;
      capelin_called_cap = 0;
      redfish_called_red = 0;
      redfish_called_cod = 0;
   redfish_called_cap = 0;

      M_cod = mean(training_cod(:,keep1));
      M_capelin = mean(training_capelin(:,keep1));
   M_red = mean(training_red(:,keep1));

      K_cod = cov(training_cod(:,keep1));
      K_capelin = cov(training_capelin(:,keep1));
      K_red = cov(training_red(:,keep1));

      for i = 1:length(alldata)
            data = [alldata(i,keep1)];

            dist_cod = (data - M_cod) * inv(K_cod) * (data - M_cod)';
            dist_capelin = (data - M_capelin) * inv(K_capelin) * (data -
M_capelin)';
            dist_red = (data - M_red) * inv(K_red) * (data - M_red)';
            if ( (dist_cod < dist_capelin) & (dist_cod < dist_red)  )
                  called = 1; % 1 = cod;
            elseif ( (dist_capelin < dist_cod) & (dist_capelin < dist_red))
            called = 2; % 2 = capelin
            elseif ((dist_red < dist_cod) & (dist_red < dist_capelin) )
            called = 3; % 3 = redfish
            else
            called = 4; % we have a tie
            printf('\nwe have a tie with the distances\n');
            pause;
            end
```

```
            if called == 1
            if (i <= num_cod)
             cod_called_cod = cod_called_cod + 1;
            elseif (i > num_cod) & (i <= (num_cod + num_capelin))
             capelin_called_cod = capelin_called_cod + 1;
                 else
              redfish_called_cod = redfish_called_cod + 1;
             end

            elseif called == 2
            if (i <= num_cod)
                 cod_called_cap = cod_called_cap + 1;
            elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                 capelin_called_cap = capelin_called_cap + 1;
            else
                 redfish_called_cap = redfish_called_cap + 1;
            end

            elseif called == 3
                 if (i <= num_cod)
                 cod_called_red = cod_called_red + 1;
            elseif (i > num_cod) & (i <= (num_cod + num_capelin))
                 capelin_called_red = capelin_called_red + 1;
            else
                 redfish_called_red = redfish_called_red + 1;
            end
            end
      end

      fprintf('Number of Features = %d\nFeature List = ',length(keep1));
      fprintf('%d  ', keep1);
      fprintf('\n\t\t\t\tPREDICTED CLASS\n');
      fprintf('\t\tCOD\t\tCAPELIN\t\tREDFISH\n');

      fprintf('COD\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n',...
      cod_called_cod,cod_called_cod/num_cod*100,...
      cod_called_cap,cod_called_cap/num_cod*100,...
      cod_called_red,cod_called_red/num_cod*100);

      fprintf('CAPELIN\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n',...
      capelin_called_cod,capelin_called_cod/num_capelin*100,...
      capelin_called_cap,capelin_called_cap/num_capelin*100,...
      capelin_called_red,capelin_called_red/num_capelin*100);

      fprintf('REDFISH\t\t%.0f [%.1f %%]\t%.0f [%.1f %%]\t%.0f [%.1f %%]\n\n',...
      redfish_called_cod,redfish_called_cod/num_redfish*100,...
      redfish_called_cap,redfish_called_cap/num_redfish*100,...
      redfish_called_red,redfish_called_red/num_redfish*100);

%     perf = (cod_called_cod/num_cod) + (capelin_called_cap/num_capelin) +
(redfish_called_red/num_redfish),

end

% temp to illustrate overtraining
plot(perf_per_num_feats)
```

BIBLIOGRAPHY

Albers, V. M. (1965). *Underwater Acoustics Handbook - II*, The Pennsylvania State University Press, University Park, Pennsylvania, USA, 356 p.

Bondarenko, V. M., Gavrilov, E. N., and Tarasov, S.P. (1990). "Use of Parametric Transducers for Wideband Measurement of Fish Target Strength", *Rapports et Proces-Verbaux, Reun. Cons. int. Explor.* Vol.189: pp. 366-369.

Burczynski, J. (1982). "Introduction to the Use of Sonar Systems for Estimating Fish Biomass," FAO Fisheries Technical Paper, No. 191, Rev. 1, Food and Agriculture Organization of the United Nations, Rome, 89 p.

Cobra, D. T. and de Moraes, H. A. (1994). "Classification of Side-Scan Sonar Images Through Parametric Modelling," *Proceedings of Oceans'94*, Brest, France. Vol. 2, pp. 461-464.

Craig, R. E. (1979). "Units, Symbols, and Definitions in Fisheries Acoustics," *Meeting on Hydroacoustical Methods for the Estimation of Marine Fish Populations*, The Charles Stark Laboratory Inc., Cambridge, Mass. Vol. 2, part A, pp. 23-52.

Diachok, O. (1994). "Modal Attenuation Due to Fish," *Proceedings of the Second European Conference on Underwater Acoustics*, Lyngby, Denmark. Vol. 1, pp. 331-336.

Foote, K. G., Knudsen, H. P., Vestnes, G., MacLennan, D. N., and Simmonds, E. J. (1987). "Calibration of Acoustic Instruments for Fish Density Estimation: A Practical Guide," International Council for the Exploration of the Sea Cooperative Research Report, Copenhagen, Denmark.

Furusawa, M., Ishii, K., and Miyanohana, Y. (1992). "Attenuation of Sound by Schooling Fish," *Acoustical Society of America Journal*, Vol. 92, pt. 1, pp. 987-994.

Hackman, R. H., and Specht, D. F. (1992). "An Application of the PNN Algorithm to the Active Classification of Sonar Targets," *Proceedings of Oceans '92*, Newport, Rhode Island, USA. Vol. 1, pp. 141-146.

Jain, A. K. (1989). *Fundamentals of Digital Signal Processing*, Prentice Hall, Englewood Cliffs, New Jersey, USA, 569 p.

Kubecka, J., Duncan, A. and Butterworth, A. (1992). "Echo Counting or Echo Integration for Fish Biomass Assessment in Shallow Waters," *Proceedings of the European Conference on Underwater Acoustics*, Luxembourg, pp. 129-132.

MacLennan, D.N. and Forbes, S.T. (1984). "Fisheries Acoustics: A Review of General Principles," *Rapports et Proces-Verbaux, Reun. Cons. int. Explor*, Vol. 184, pp. 7-

Misund, O. A., Totland, B., Floen, S., and Aglen, A. (1994). "Computer Based Detection of Schools by Multi-beam Sonar," *Proceedings of the Second European Conference on Underwater Acoustics*, Lyngby, Denmark. Vol. 2, pp. 815-820.

Scalabrin, C., Wiell, A. and Diner, N. (1992). "The Structure of Multidimensional Data from Acoustic Detection of Fish Schools," *Proceedings of the European Conference on Underwater Acoustics*, Luxembourg, pp. 141-146.

Scalabrin, C., and Lurton, X. (1994). "Fish Shoals Echo Amplitude Analysis," *Proceedings of the Second European Conference on Underwater Acoustics*, Lyngby, Denmark. Vol. 2, pp. 807-814.

Sun, Z., and Gimenez, G. (1994). "Influence of Target Composition on the Relationship Between Echo Energy and Target Quantity," *Acoustical Society of America Journal*, Vol. 96, part 1, no.5, pp. 3080-3087.

Sun, Z. and G. Gimenez. (1994). "Influence of the Time Varied Gain Amplification on the Validity of the Echo Integration Method in the Estimation of Underwater Target Abundance," *Proceedings of the Second European Conference on Underwater Acoustics*, Lyngby, Denmark, July 1994, Vol. 2, pp. 807-814.

Therrien, C.W. (1989). *Decision, Estimation and Classification: An Introduction to Pattern Recognition and Related Topics*, John Wiley & Sons Inc., New York, New York, USA.

Yudanov, K.I. and I.L. Kalikhman. (1979). "Sound Scattering by Marine Animals," *Meeting on Hydroacoustical Methods for the Estimation of Marine Fish Populations*, The Charles Stark Laboratory Inc., Cambridge, Massachusetts, June 1979. Vol. 2, Part A, pp. 53-93.